

Cryptology with CrypTool

Version 1.4.00

**Introduction to Cryptography and Cryptanalysis
Scope, Technology and Future of CrypTool**



www.cryptool.de
www.cryptool.com
www.cryptool.org

Bernhard Esslinger and CrypTool team, 2006

Content

I. CrypTool and Cryptography – Overview

1. [The CrypTool project](#)
2. [Relevance of cryptography and examples of classical encryption methods](#)
3. [Insights from cryptography development](#)

II. CrypTool features

1. [Overview](#)
2. [Interaction examples](#)
3. [Challenges for developers](#)

III. Examples

- | | |
|--|---|
| 1. <u>RSA encryption</u> | 9. <u>Side channel attack demo</u> |
| 2. <u>Digital signature</u> | 10. <u>RSA attack using lattice reduction</u> |
| 3. <u>Attacking RSA encryption</u> | 11. <u>Random analysis with 3-D visualisation</u> |
| 4. <u>Psion-analysis</u> | 12. <u>Secret Sharing with CRT</u> |
| 5. <u>Weak DES-keys</u> | 13. <u>Implementation of CRT in Astronomy</u> |
| 6. <u>Discover NSA keys</u> | 14. <u>Visualisation of encryption using ANIMAL</u> |
| 7. <u>Locate hash collisions</u> | 15. <u>Generation of a MAC</u> |
| 8. <u>Types of authentication</u> | 16. <u>Hash-Demo</u> |

IV. Project / Outlook / Contact



Content

CrypTool and Cryptography – Overview

[CrypTool features](#)

[Examples](#)

[Project / Outlook / Contact](#)

The CrypTool Project

- Origin in awareness program of a bank (in-firm training)
→ **Awareness for employees**
- Developed in Cooperation with Universities (improving education)
→ **media didactic**

1998 – **Project start** – effort more than 15 man-years

2000 – CrypTool available as **freeware**

2002 – CrypTool on **Bürger-CD-ROM from BSI** „Ins Internet – mit Sicherheit“

2003 – CrypTool becomes **Open-Source** – Hosting through University of Darmstadt
(Prof. Eckert)

2004 – Awards

TeleTrust (TTT Förderpreis 2004)

NRW (IT Security Award NRW)

RSA Europe (Finalist of European Information Security Award 2004)



Ministerium für Innovation,
Wissenschaft, Forschung
und Technologie des Landes
Nordrhein-Westfalen

NRW.



▪ Developers

- Developed by People from different Companies and Universities
- Additional project members or usable sources are always appreciated (up to now there are around 30 people working on CrypTool world wide).

Relevance of Cryptography

Typical Scenarios for using Cryptography in the daily life

Examples for Cryptography Usage

- Phone cards, cell phones, remote controls
- Cash machines, money transfer between banks
- Electronic cash, online banking, secure eMail
- Satellite TV, Pay TV
- Immobiliser systems in cars
- Digital Rights Management (DRM)



- Cryptography is no longer limited to agents, diplomats or the military. Cryptography is a modern, mathematically characterised science.
- Breakthrough for cryptography started with the broad use of the Internet
- For companies and governments it is important that systems are secure and ...
... users (clients, employees) have a certain understanding and awareness for IT security!



Cryptography – Objectives

Protection goals related to Cryptography

- **Confidentiality**

- *Information is can practically not made available or disclosed to unauthorized individuals, entities or processes.*

- **Authentication**

- *Authentication ensures that users are identified and those identities appropriately verified.*

- **Integrity**

- *Integrity ensures that data has not been altered or destroyed in an unauthorized manner.*

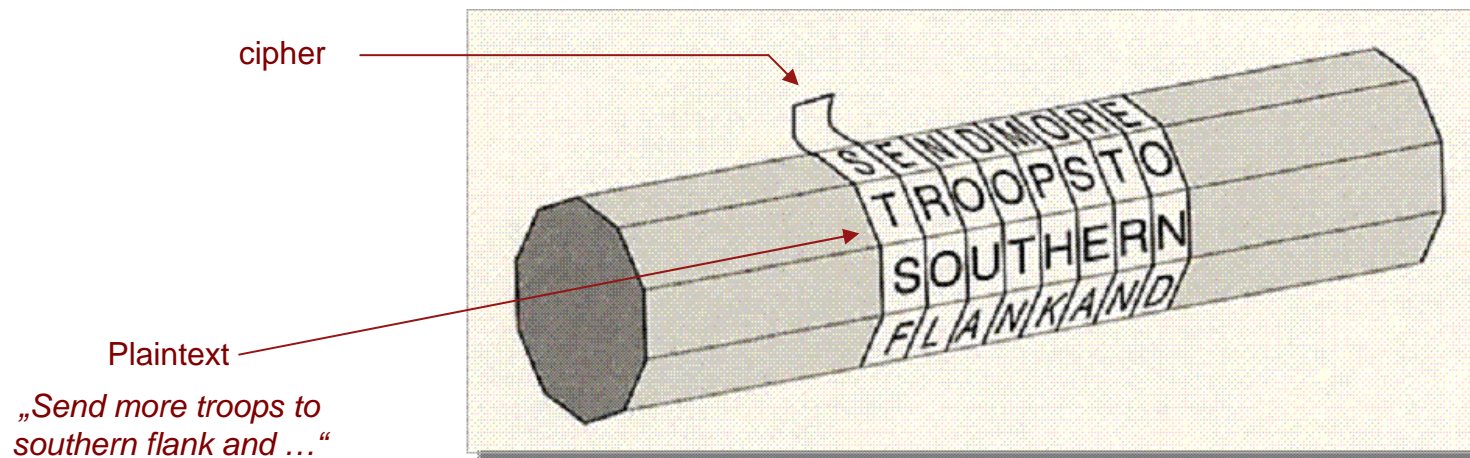
- **Non-Repudiation**

- *The principle that, afterwards, it can be proven that the participants of a transaction did really authorize the transaction and that they have no means to deny their participation.*

Examples of early cryptography (I)

Ancient encryption methods

- **Tattoo on a slave's head concealed by re-grown hair**
- **Atbash** (around 600 B.C.)
 - Hebrew secret language, reversed alphabet
- **Skytale from Sparta** (500 B.C.)
 - Described by Greek historian/author Plutarch (45 - 125 AD)
 - Two cylinders (wooden rod) with identical diameter
 - Transposition (plaintext characters are re-sorted)



Examples of early cryptography (II)

Symmetric Caesar encryption

- **Caesar encryption** (Julius Cäsar, 100 - 44 v.Chr.)
- Simple substitution cipher

GALLIA EST OMNIS DIVISA ...

Plaintext: ABCDEF**G**H I J K L M N O P Q R S T U V W X Y Z

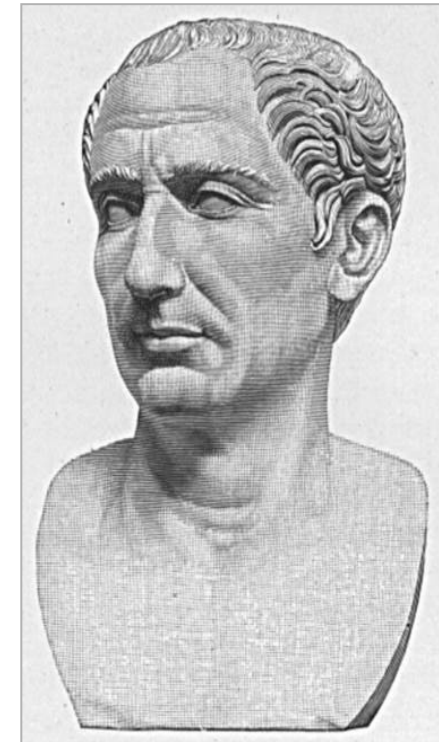
Secret alphabet: DEFGH I **J** K L M N O P Q R S T U V W X Y Z A B C

JDOOLD HVW RPQLV GLYLVD ...

- **Attack:** Frequency-analysis (typical character allocation)

Presentation with CrypTool via the following menus:

- **Animation:** “Indiv. Procedures” \ “Visualization of algorithms using ANIMAL” \ “Caesar ...”
- **Implementation:** “Crypt/Decrypt” \ “Symmetric (classic)” \ “Caesar / Rot 13”



Examples of early cryptography (II)

Symmetric Vigenère encryption

- **Vigenère Encryption** (Blaise de Vigenère, 1523-1596 AD)
- Encryption with a key word using a key table
- Key word: **CHIFFRE**
- Encrypting: **VIGENERE** becomes **XPOJSVVG**
- The plaintext character (V) is replaced by the character in the corresponding row and in the column of the first key word character (C). The next plaintext character (I) is replaced by the character in the corresponding row and in the column of the next key word character (H), and so on.
- If all characters of the key word have been used, then the next key word character is the first key character.
- **Attack** (via Kasiski test): Plaintext combinations with an identical cipher text combination can occur. The distance of these patterns can be used to determine the length of the keyword. An additional frequency analysis can then be used to determine the key.

Keyword character

The diagram shows a 26x26 Vigenère square. A red arrow points from the 'Keyword character' label to the letter 'C' in the first row and third column. Another red arrow points from the 'Plaintext character' label to the letter 'V' in the 21st row and first column. A third red arrow points from the 'Encrypted character' label to the letter 'X' in the 21st row and third column. A dashed line connects 'V' and 'X' through the 'C'. The square is labeled 'Tableau carré, dit « Carré de Vigenère »' at the bottom.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tableau carré, dit « Carré de Vigenère »

Plaintext character

Encrypted character

Examples of early cryptography (IV)

Other symmetric encryption methods

- **Homophone substitution**
- **Playfair** (1854 Sir Charles Wheatstone, 1802-1875)
 - published by Baron Lyon Playfair
 - Substitution of a character pair by another based on a square-based alphabet array
- **Transfer of book pages**
 - Adaptation of the One-Time-Pad (OTP)
- **Hole template** (Fleißner)
- **Permutation encryption** („Double Dice")
 - Transposition / very effective

Key Entry: Playfair

Optionen

- ☒ Pre-format text
- ☒ Ignore duplicates within the key phrase

Playfair key

Short version of the Playfair key:

CHARLES

Key matrix

C	H	A	R	L	
E	S	B	D	F	
V	W	X	Y	Z	
G	I	K	M	N	
O	P	Q	T	U	

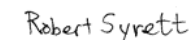
☒ 5x5 Matrix
☐ 6x6 Matrix

Encrypt Decrypt Cancel

Cryptography developments in the last 100 years

- ... are still in use today.
(since, not all can be done by a computer...)
- ... and their principals of **transposition** and **substitution** are inputs for the design of modern algorithms:
combinations of simple operations (a type of multiple encryption, a so called cascades of ciphers), on bit level, block cipher, rounds.

- ... more sophisticated,
- mechanised or computerised and
- remains symmetric.



Examples of the first half of the 20th century

Mechanical encryption machines (rotor machines)

- **Enigma Encryption** (Arthur Scherbius, 1878-1929)
- More than 200.000 machines have been used in WW2
- The rotating cylinder set causes, that every character of the text becomes encrypted with a new permutation.
- Broken by massive effort of cryptography experts (around 7.000 person in UK) with decryption machines, captured original Enigmas and by intercepting daily status reports (e.g. weather reports).

- **Consequences of this successful crypto analysis:**

“In general the successful crypto analysis of the engima encryption has been a strategic advantage, that has played a significant role in winning the war. Some historians assume that the break of the enigma code has shorten the war by several months or even a year.”

(translated from http://de.wikipedia.org/wiki/Enigma_%28Machine%29 - 6 March 2006)



Cryptography – Important Insights (I)

- **Kerckhoffs principle** (1883)

- Separation of algorithm (method) and key

Algorithm: “Shift alphabet by a certain number of positions to the left”

Key: the “certain number of positions” (Caesar for example)

- Kerckhoffs principle: The secret lies within the key and not within the algorithm or „No security through obscurity“

- **One Time Pad – Shannon / Vernam**

- Demonstrably theoretically secure, but not usable in reality (only red phone)

- **Shannons concepts: Confusion and Diffusion**

- Relation between M, C and K has to be as complex as possible
- Every cipher text character should depend on as many plaintext characters and as many character of encryption key as possible
- „Avalanche effect“ (small modification, big impact)

- **Trapdoor function** (one-way function)

- Fast in one direction, very slow in the opposite direction
- Only the secret key grants access to trapdoor



Examples for a breach of the Kerckhoffs principle

Secret lies within the key and not within the algorithm

- **Cell phone encryption penetrated** (December 1999)

„Israeli researchers have discovered design flaws that allow the descrambling of supposedly private conversations carried by hundreds of millions of wireless phones. Alex Biryukov and Adi Shamir describe in a paper to be published this week how a PC with 128 MB RAM and large hard drives can penetrate the security of a phone call or data transmission in less than one second. The flawed algorithm appears in digital GSM phones made by companies such as Motorola, Ericsson, and Siemens, and used by well over 100 million customers in Europe and the United States.” [...]

*“Previously the GSM encryption algorithms have come under fire for **being developed in secret away from public scrutiny** -- but most experts say high security can only come from published code. Moran said "it wasn't the attitude at the time to publish algorithms" when the A5 ciphers was developed in 1989, but **current ones being created will be published for peer review.**”*

[<http://wired.lycos.com/news/politics/0,1283,32900,00.html>]

Sample of a One Time Pad Adaptation



Clothes hanger of a Stasi agent
with a secret One Time Pad
(taken from: *Spiegel Spezial* 1/1990)

Key Distribution Problem

Key distribution for a symmetric encryption

If **2 persons** communicate with each other using symmetric encryption, they **need one secret key**.

If n persons communicate with each other, then they need $S_n = n(n-1) / 2$ keys.

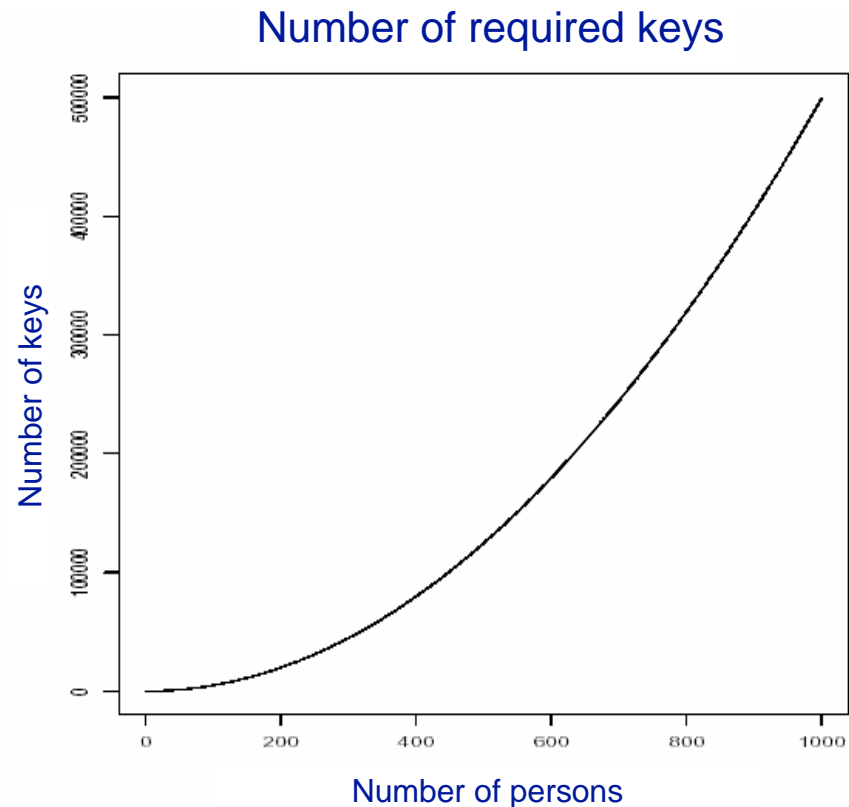
This means

$n = 100$ persons require

$S_{100} = 4.950$ keys, and

$n = 1.000$ persons require

$S_{1000} = 499.500$ keys.



Cryptography – Important Insights (II)

Solving the key distribution problem through asymmetric cryptography

- **Asymmetric cryptography**

- **For centuries it was believed that:** Sender and receiver need same secret.
- **New:** Every member needs a key pair (Solution of the key distribution problem)

- **Asymmetric encryption**

- „Everyone can lock a padlock or can drop a letter in a mail box.“
- MIT, 1977: Leonard Adleman, Ron Rivest, Adi Shamir (well known as RSA)
- GCHQ Cheltenham, 1973: James Ellis, Clifford Cocks (admitted in public December 1997)

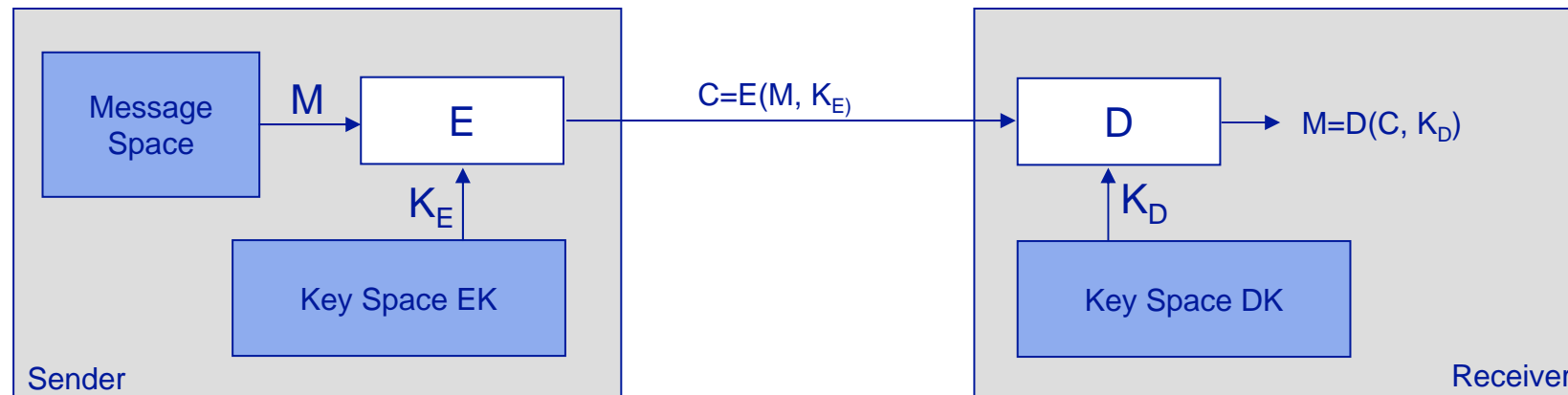
- **Key distribution**

- Stanford, 1976: Whitfield Diffie, Martin Hellman, Ralph Merkle (Diffie-Hellman Key Exchange)
- GCHQ Cheltenham, 1975: Malcolm Williamson

Security in open networks (such as the internet) would be extremely expensive and complex without asymmetric cryptography!

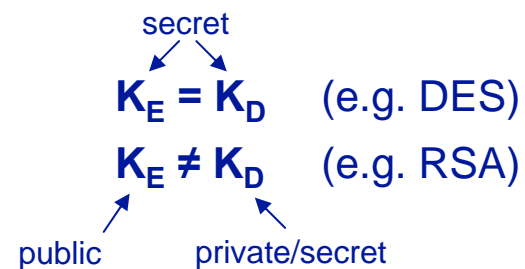
Encryption and Decryption

Symmetric und asymmetric encryption



a) Symmetric Encryption:

b) Asymmetric Encryption:



Cryptography – Important Insights (III)

Increasing relevance of mathematics and information technology

- **Modern cryptography** is based on **mathematics**
 - Still new symmetric encryption methods such as AES (better performance and shorter key length compared to the asymmetric methods purely based on mathematical problems).
- The security of encryption methods heavily depends on the current status of **mathematics** and **information technology** (IT)
 - Computation complexity (meaning processing effort in relation to key length, storage demand and data complexity)
 - > see RSA: Bernstein, TWIRL-Device, RSA-160
 - Very high activity in current research:
Factorisation, non-parallelize algorithm (because of quantum computing), better understanding of protocol weaknesses and random generators, ...).
- Serious mistake: *“Real mathematics has no effects on the war.”* (G.H. Hardy, 1940)
- Vendors discover **security** as an essential **purchase criterion**

Demo in Cryptool

- *Statistical Analysis*

- *Encrypting twice is not always better:*

Caesar: $C + D = G$ ($3 + 4 = 7$)

Vigenère: - CAT + DOG = FOZ [(2,0,19)+(3,14,6)=(5,14,25)]

- "Hund" + "Katze" = "RUGCLENWGYXDATRNNHNMH")

- *Vernam (OTP)*

- *AES (output key, brute-force analysis)*



Content

[CrypTool and Cryptography – Overview](#)

CrypTool features

[Examples](#)

[Project / Outlook / Contact](#)

CrypTool Features

E-Learning

1. What is CrypTool?

- Freeware program with graphical interface
- Cryptographic methods can be applied and analysed
- Comprehensive online help (understandable without deeper cryptography knowledge)
- Contains nearly all state-of-the-art cryptography functions
- Easy entry into modern and classical cryptography
- Not a “*hacker tool*”

2. Why CrypTool?

- Origin in awareness initiative of a commercial bank
- Developed in close cooperation with universities
- Improvement of university education and in-firm training

3. Target Group

- *Core group*: Students of computer science, business computing and mathematics
- *But also for*: computer users, application developers, employees
- *Prerequisite*: PC knowledge
- *Preferable*: Interest in mathematics and/or programming

Content of the program package

CrypTool Program

- All functions integrated in a *single* program with consistent graphical interface
- Runs on Win32
- Cryptography libraries from Secude and OpenSSL
- Long integer arithmetic from Miracl and GMP, Lattice base reduction via NTL (Shoup)

AES-Tool

- Standalone program for AES encryption (self extracting)

Educational game

- „Numbers Shark“ encourages the understanding of factors and prime numbers.

Comprehensive Online Help (HTML-Help)

- Context-sensitive help available via F1 for *all* program functions (including menus)
- Detailed use cases for a lot of program functions (tutorial)

Script (.pdf file) with background information

- Encryption methods • Prime factorisation • Digital signature
- Elliptic curves • public key certification • Basic number theory

Two short stories related to cryptography by Dr. C. Elsner

- „The Dialogue of the Sisters“ (a RSA variant as key element)
- „The Chinese Labyrinth“ (Numbers theory tasks for Marco Polo)

Features (I)

Cryptography

Classical cryptography

- Caesar
- Vigenère
- Hill
- Homophone Substitution
- Playfair
- ADFGVX
- Addition
- XOR
- Vernam
- Permutation
- Solitaire

Several options to easily understand the cryptography methods

- Selectable alphabet
- Options: handling of blanks, etc.

Crypto analysis

Attack on classical methods

- Cipher text Only
 - Caesar
 - Vigenère
 - Addition
 - XOR
 - Substitution
 - Playfair
- Known Plaintext
 - Hill
- Manual
 - Mono alphabetical substitution
 - Playfair
 - Solitaire

Supported analysis methods

- Entropy, floating frequency
- Histogram, N-Gram-Analysis
- Autocorrelation
- Periodicity

Features (II)

Cryptography

Modern symmetric Encryption

- IDEA, RC2, RC4, RC6, DES, 3DES
- Serpent, Twofish
- AES candidates of the last selection round
- AES (=Rijndael)

Asymmetric Encryption

- RSA with X.509-Certificates
- RSA-Demo
 - Understanding of examples
 - Selectable alphabet and block length

Hybrid Encryption (RSA + AES)

- Interactive data flow diagram

Crypto analysis

Brute-force-Attack on symmetric algorithm

- For all algorithms
- Assumption: Entropy of plaintext is small

Attack on RSA encryption

- Factorisation of RSA-module
- Lattice-Based attacks

Attack on hybrid encryption

- Attack on RSA or
- Attack on AES (side-channel attack)

Features (III)

Cryptography

Digital signature

- RSA with X.509-Certificates
 - Signature as interactive data flow diagram
- DSA with X.509-Certificates
- Elliptic Curve DSA, Nyberg-Rueppel

Hash functions

- MD2, MD4, MD5
- SHA, SHA-1, RIPEMD-160

Random generators

- Secude
- $x^2 \bmod n$
- Linear congruence generator (LCG)
- Inverse congruence generator (ICG)

Crypto analysis

Attack on RSA signature

- Factorisation of the RSA-module
- feasible up to 250 bits or 75 decimal places (on standard desktop PCs)

Attack on Hash functions / digital signature

- Generate Hash collisions for ASCII based text (birthday paradox) (up to 40 bit in around 5 min)

Analysis of random data

- FIPS-PUB-140-1 Test-Battery
- Periodicity, Vitany, Entropy
- Floating frequency, histogram
- n-Gram-Analysis, autocorrelation
- ZIP compression test

Features (IV)

Animation / Demos

- Caesar, Vigenère, Nihilist, DES with ANIMAL
- Hybrid encryption and decryption
- Generation and verification of digital signatures
- Diffie-Hellman key exchange
- Secret Sharing (with CRT or Shamir)
- Challenge-Response method (authentication)
- Side channel attack
- Graphical 3D presentation of (random) data streams
- Sensitivity of hash functions regarding plaintext modifications
- Numbers theory and RSA crypto system



Features (V)

Additional Functions

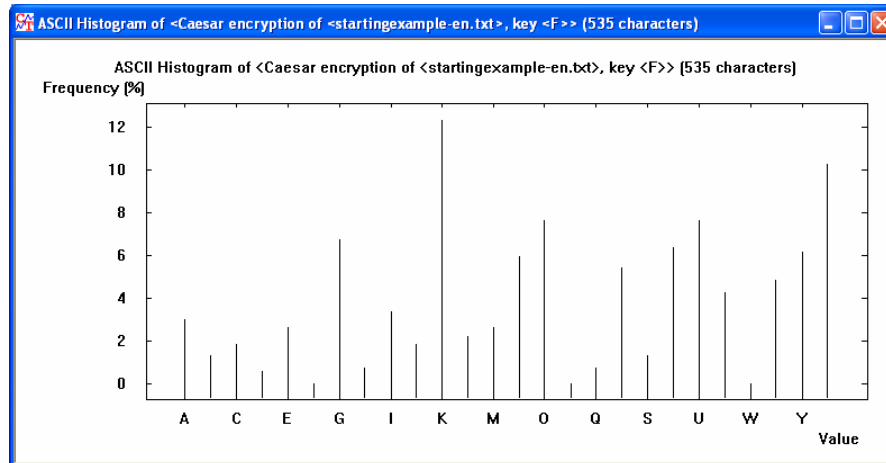
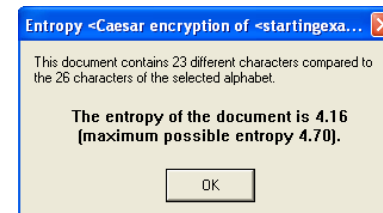
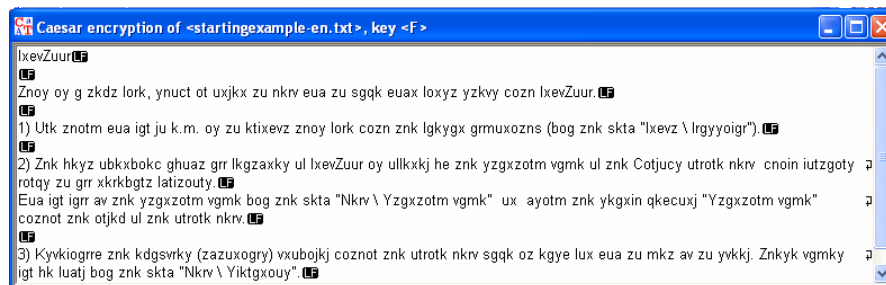
- Homophone and permutation encryption
- PKCS #12 import and export for PSEs (Personal Security Environment)
- Generate hashes of large files, without loading them
- Brute force attacks on symmetric algorithms
- ...

Language structure analysis

Language analysis options available in CryptTool

Number of characters, n-Gram, Entropy

- e.g. using CryptTool „Analysis / Tools for Analysis /...”



N-Gram List of Caesar encryption of <startingexample-en.txt>, key <F>

Selection

☒ Histogram

☐ Digram

☐ Trigram

☐ 4-gram

Display of the 26 most common N-Grams (allowed values: 1-5000)

Determine list

Save list

Close

No.	Charact...	Frequency in %	Frequency
1	K	12.3364	66
2	Z	10.2804	55
3	O	7.6636	41
4	U	7.6636	41
5	G	6.7290	36
6	T	6.3551	34
7	Y	6.1682	33
8	N	5.9813	32
9	R	5.4206	29
10	X	4.8598	26
11	V	4.2991	23
12	I	3.3645	18
13	A	2.9907	16
14	E	2.6168	14
15	M	2.6168	14
16	L	2.2430	12
17	C	1.8692	10
18	J	1.8692	10
19	B	1.3084	7
20	S	1.3084	7
21	H	0.7477	4
22	Q	0.7477	4
23	D	0.5607	3

Demonstration of Interactivity (I)

Vigenère analysis

Demo in CrypTool

The result of the Vigenère analysis can be manually reworked (changing the key length):

1. Encrypt starting example with **TEST**

- „Crypt/Decrypt“ \ „Symmetric (classic)“ \ „Vigenère...”
- Enter TEST „Encrypt“

Analysis of the encryption

- „Analysis“ \ „Symmetric Encryption (classic)“ \ „Ciphertext only“ \ „Vigenère“
- Derived key length 4, Derived key TEST ✓

2. Encrypt starting example with **TESTETE**

- „Crypt/Decrypt“ \ „Symmetric (classic)“ \ „Vigenère...”
- Enter TESTETE „Encrypt“

Analysis of the encryption

- „Analysis“ \ „Symmetric Encryption (classic)“ \ „Ciphertext only“ \ „Vigenère“
- Derived key length 5 – not correct ✗
- Key length manually set to 7
- Derived key TESTETE ✓

Demonstration of Interactivity (II)

Automated factorisation

Demo in CrypTool

Factorisation of a compound number with factorisation algorithms

- „Indiv. Procedures“ \ „RSA Cryptosystem“ \ „Factorisation of a Number“
- Some methods are executed in parallel (multi threaded)
- Methods have specific advantages and disadvantages (e.g. some methods can only determine small factors)

Factorisation example 1:

316775895367314538931177095642205088158145887517

48-digit decimal number

=

3 * 1129 * 6353 * 1159777 * 22383173213963 * 567102977853788110597

Factorisation example 2:

$2^{250} - 1$

75-digit decimal number

=

3 * 11 * 31 * 251 * 601 * 1801 * 4051 * 229668251 * 269089806001 *
4710883168879506001 * 5519485418336288303251

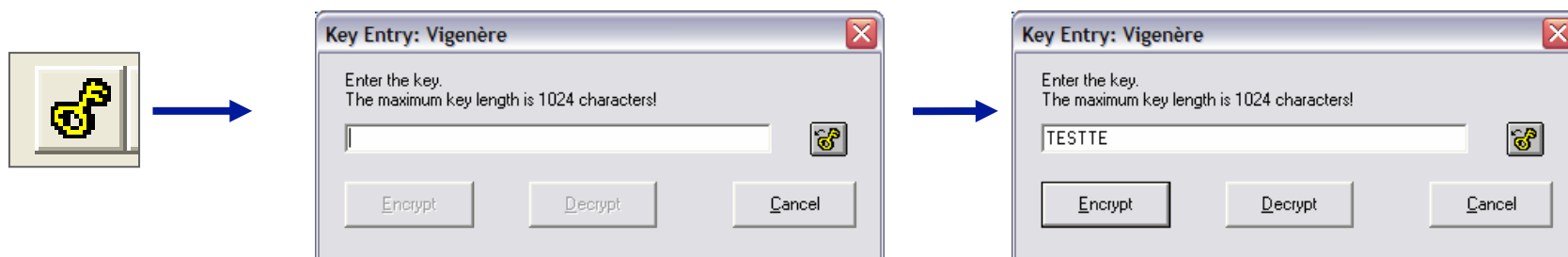
Concepts for a user-friendly Interface

1. Context sensitive help (F1)

- F1 on a selected menu entry shows information about the algorithm/method.
- F1 in a dialog box explains the use of the dialog.
- These assistances and the contents of the superordinate menus are cross linked in the online help.

2. Paste of keys in key-input dialog

- CTRL-V can be used to paste contents from the clipboard.
- Used keys can be taken out of cipher text windows via an icon in the icon bar. A corresponding icon in the key-input dialog can be used to paste the key into the key field. A CrypTool-internal memory which is available for every method is used (helpful for „specific“ keys – e.g. homophone encryption).



Challenges for Developers (Examples)

1. Many functions running parallel

- Factorisation runs with multi threaded algorithms

2. High performance

- Birthday paradox to locate hash collisions or for brute force analysis

3. Consider memory limits

- Floyd-algorithm (mappings to locate hash collisions) or Quadratic Sieve

4. Time measurement and estimates

- Display of elapsed time while using brute force

5. Reusability / Integration

- Forms for prime number generation
- RSA cryptosystem (switches after successful attack from PubKey view to PrivKey user)



Content

[CrypTool and Cryptography – Overview](#)

[CrypTool features](#)

Examples

[Project / Outlook / Contact](#)

CrypTool Examples

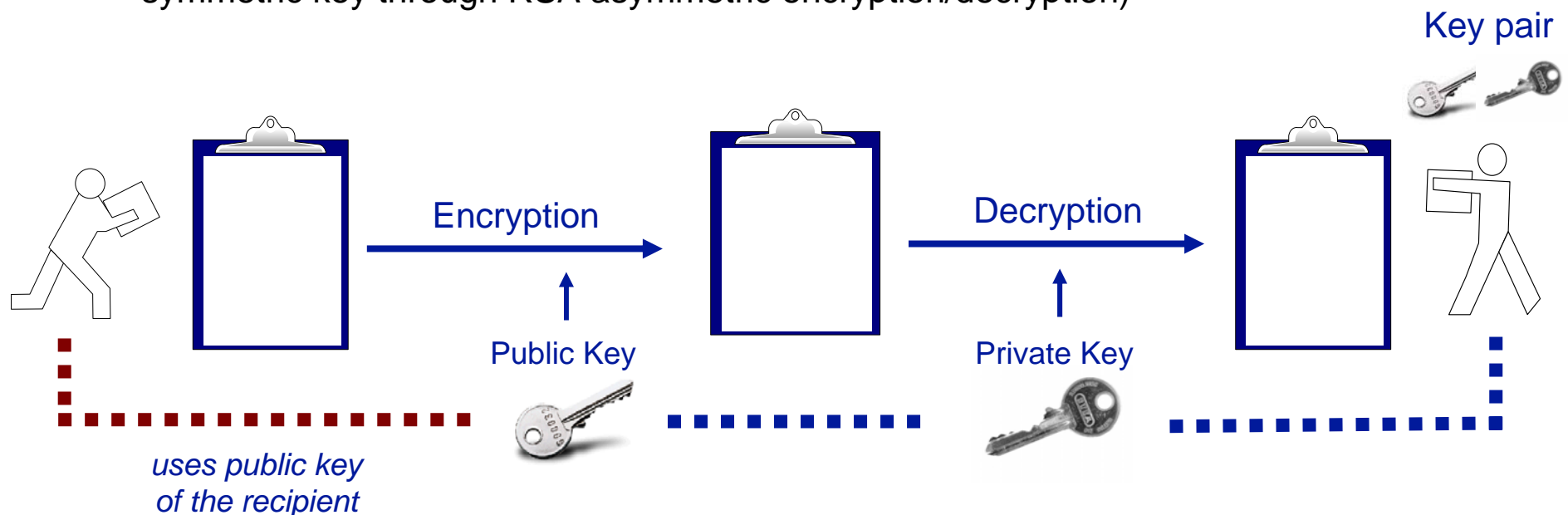
Overview of Examples

1. [Encryption with RSA / Prime number test / Hybrid encryption and digital certificates](#)
2. [Digital signature visualised](#)
3. [Attack on RSA encryption \(modul N too short\)](#)
4. [Analysis of encryption in PSION 5](#)
5. [Weak DES keys](#)
6. [Locating key material \("NSA-Key"\)](#)
7. [Attack on digital signature through location of hash collision](#)
8. [Authentication in a client-server environment](#)
9. [Demonstration of a side channel attack \(on hybrid encryption protocol\)](#)
10. [Attack on RSA using lattice reduction](#)
11. [Random analysis with 3-D visualisation](#)
12. [Secret Sharing \(Chinese Remainder Theorem \(CRT\) / Shamir\)](#)
13. [Implementation of CRT in Astronomy](#)
14. [Visualisation of symmetric encryption methods using ANIMAL](#)
15. [Generation of a message authentication code \(MAC\)](#)
16. [Hash Demo](#)

Examples (I)

Encryption with RSA (in reality mostly hybrid encryption)

- **Basis** for e.g. SSL protocol (access to protected web sites)
- **Asymmetric encryption using RSA**
 - Every user has a key pair – one public and one private key
 - Sender encrypts with public key of the recipient
 - Recipient decrypts with his private key
- Implemented usually in a combination with symmetric methods (transfer of the symmetric key through RSA asymmetric encryption/decryption)



Examples (I)

Encryption using RSA – Mathematical background / algorithm

- Public key: (n, e)
- Private key: (d)

where:

p, q large, randomly chosen prime numbers with $n = p \cdot q$;

d is calculated under the constraints $\gcd[\varphi(n), e] = 1$; $e \cdot d \equiv 1 \pmod{\varphi(n)}$.

Encryption and decryption operation: $(m^e)^d \equiv m \pmod{n}$

- n is the module, which length in bits is referred to as RSA key length.
- \gcd = greatest common divisor.
- $\varphi(n)$ is the Euler phi function.

Procedure:

- Transformation of message in binary representation
- Encrypt message $m = m_1, \dots, m_k$ block wise, with for all m_j :
 $0 \leq m_j < n$; maximum block size r , so that: $2^r \leq n$

Examples (I)

Prime number tests – For RSA huge primes are needed.

- Fast probabilistic tests
- Deterministic tests

The prime number test methods can test much faster whether a big number is prime, than the known factorization methods can divide a number of a similar size in its prime factors.

For the AKS method the GMP library (**GNU Multiple Precision Arithmetic Library**) is integrated into CrypTool.

The screenshot shows a window titled "Prime Number Test" with a close button in the top right corner. The window contains the following elements:

- Text:** "There are various methods to check if a number is a prime number (mathematician also say, to check if a number is prime). Usually probabilistic methods are applied: They are very fast, but can only determine with a certain (adjustable small) amount of probability if a number is prime. Besides that there are also deterministic methods: A provided result is of 100 % correctness (from the mathematic point of view)." This text is enclosed in a box with a light yellow background.
- Algorithms for prime number test:** A section with four radio buttons:
 - ☒ Miller-Rabin Test
 - ☐ Fermat Test
 - ☐ Solovay-Strassen Test
 - ☐ AKS Test (deterministic procedure)
- Prime Number Test:** A section containing:
 - A button labeled "Load number from file" in the top right.
 - A text input field labeled "Number to test" containing the value "2^521-1".
 - A "Result" section with a green checkmark icon and the text: "The number is a prime number: 68647976601306097149819007990813932172t".
- Buttons:** At the bottom, there are two buttons: "Test number" on the left and "Cancel" on the right.

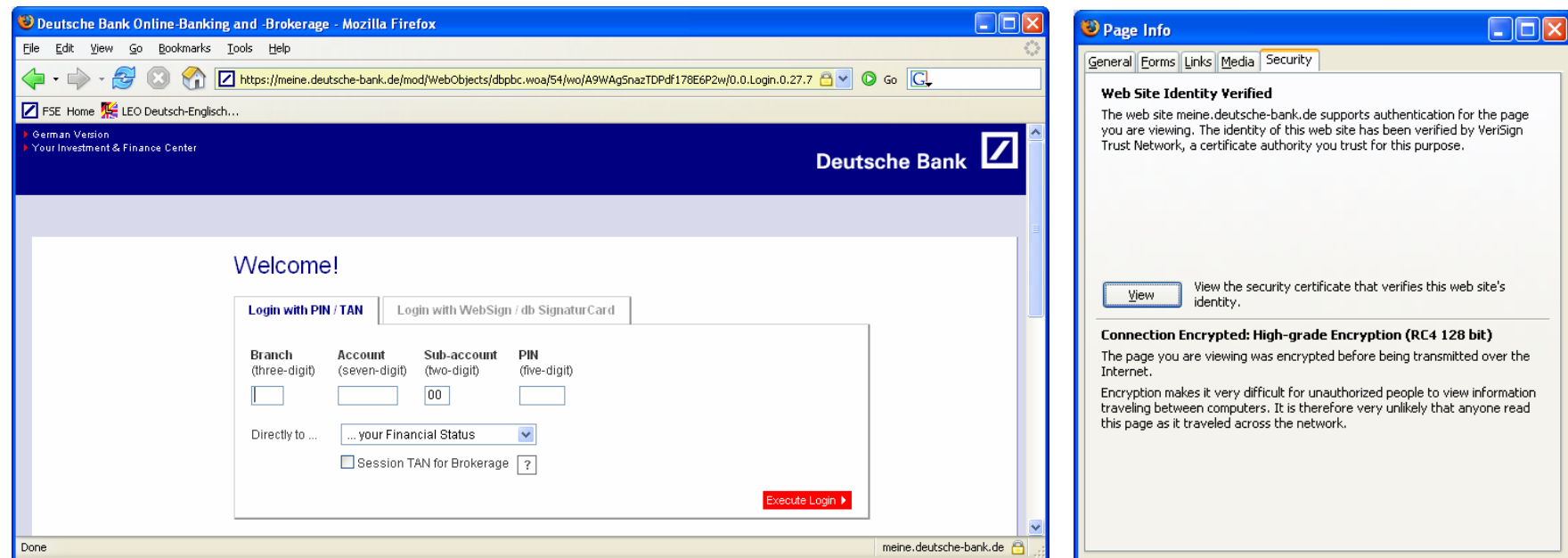
Example (I)

Hybrid encryption and digital certificates

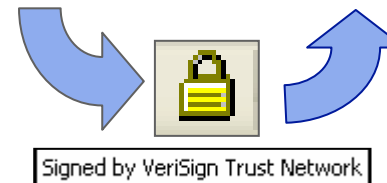
- **Hybrid encryption** – Combination of asymmetric and symmetric encryption
 1. Generation of a random symmetric key (Session Key)
 2. Session key is transferred – protected by asymmetric key
 3. Message is transferred – protected by session Key
- **Problem:** Man-in-the-middle attacks – does the public key of the recipient really belong to the recipient?
- **Solution: Digital certificates** – A central instance (e.g. VeriSign, Deutsche Bank PKI), that is being trusted by all users, ensures the authenticity of the certificate and the contained public key (similar to a passport issued by the state).
- **Hybrid encryption based on digital certificates** is the foundation for all secured electronic communication (e.g. SSL):
 - Internet Shopping and Online Banking
 - Secure eMail

Example (I)

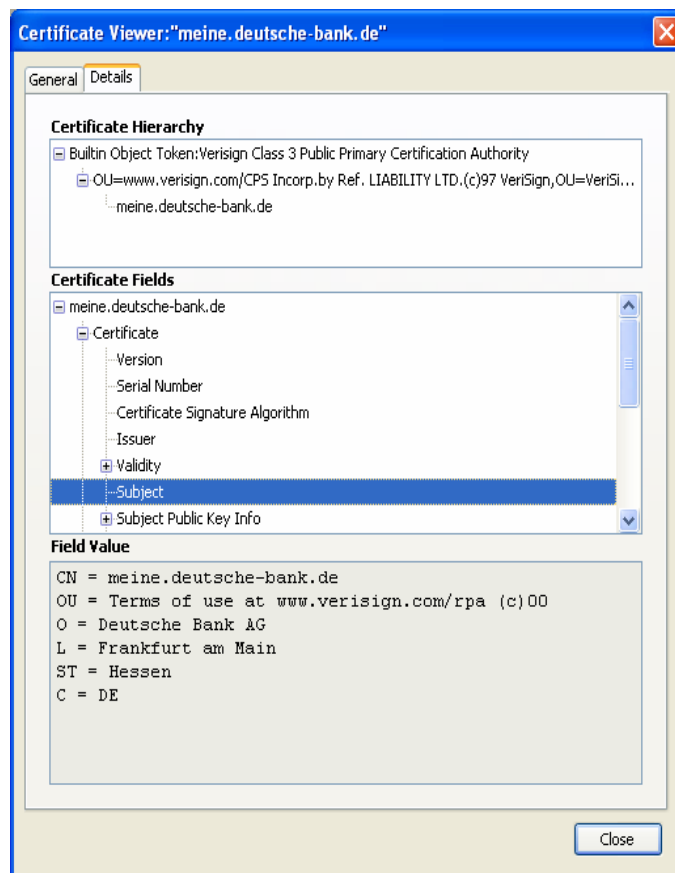
Secured online connection using SSL and certificates



This means, that the connection is authenticated (at least at one side) and that the transferred data are strongly encrypted.



Attributes or fields of a certificate



General attributes / fields

- Issuer (e.g. VeriSign)
- Requestor
- Validity period
- Serial number
- Certificate type / Version (X.509v3)
- Signature algorithm
- Public key (and method)

Public Key



Examples (II)

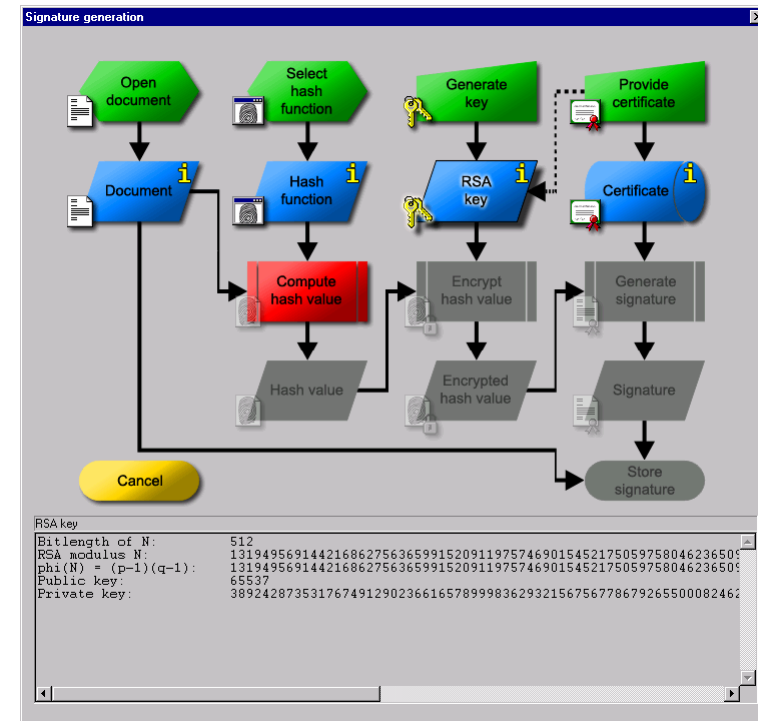
Digital signature visualised

Digital signature

- Increasingly important
 - equivalence with manual signature (digital signature law)
 - increasingly used by industry, government and consumers
- Few people know how it works

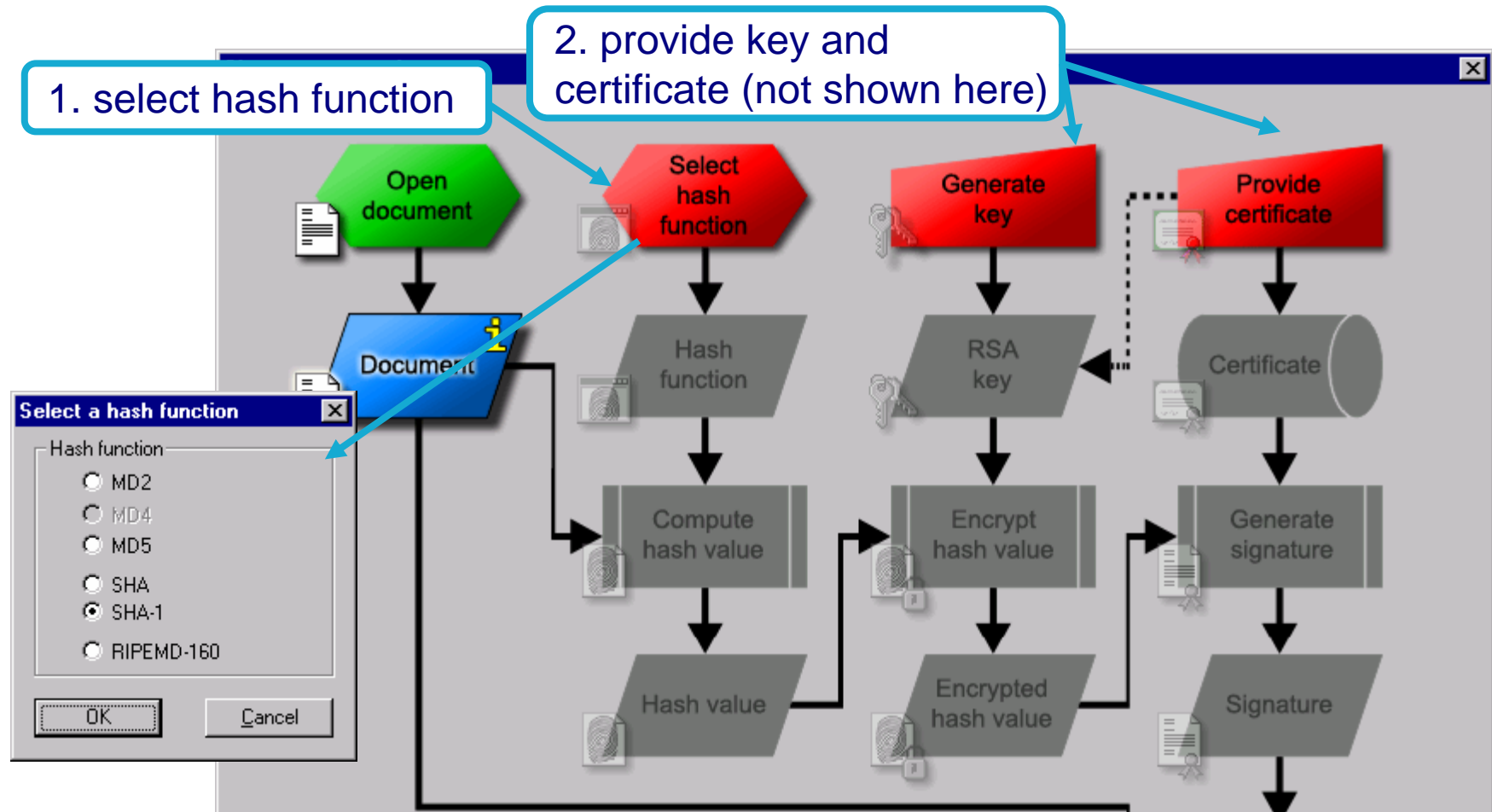
Visualisation in CrypTool

- See menu “Digital Signatures/PKI” \ “Signature Demonstration (Signature Generation)”
- Interactive data flow diagram
- Similar to the visualisation of hybrid encryption



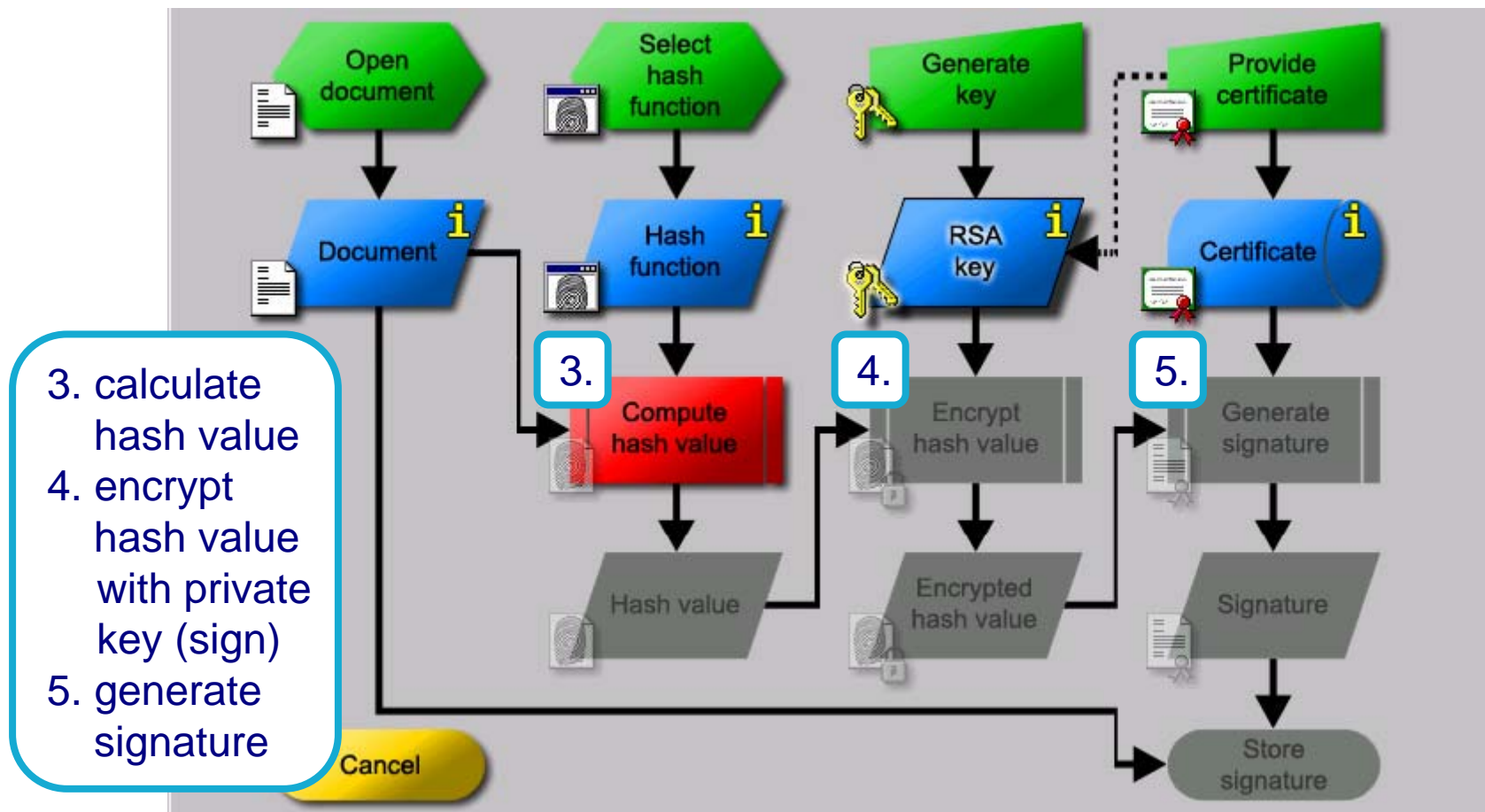
Examples (II)

Digital signature visualised: a) Preparation



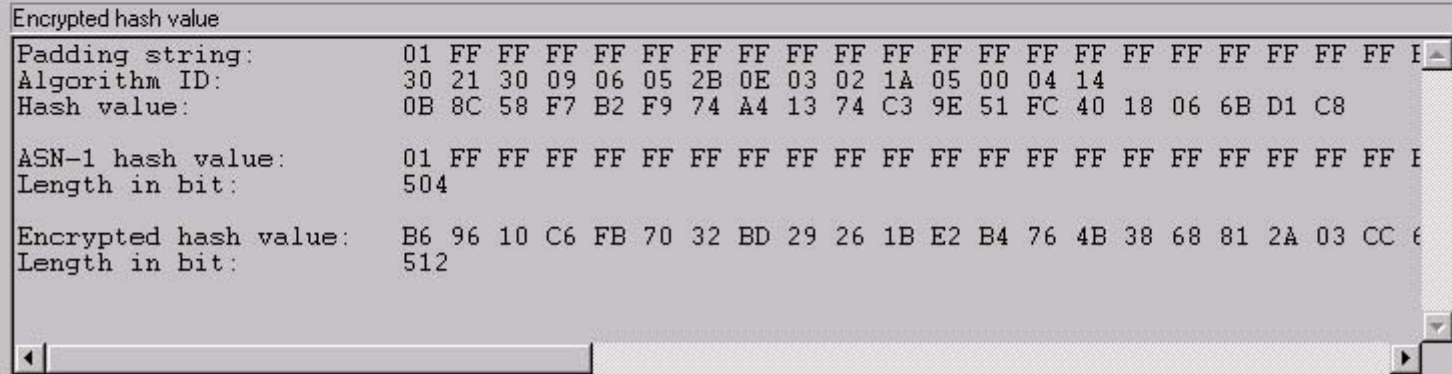
Examples (II)

Digital signature visualised: b) Cryptography



Digital signature visualised: c) Result

The operations can be performed in any order, as permitted by data dependencies.



Examples (III)

Attack on RSA encryption with short RSA modulus

Example from *Song Y. Yan, Number Theory for Computing, Springer, 2000*

- public key
 - RSA modulus $N = 63978486879527143858831415041$ (95 bit, 29 decimal digits)
 - public exponent $e = 17579$
- cipher text (block length = 14):
 - $C_1 = 45411667895024938209259253423,$
 - $C_2 = 16597091621432020076311552201,$
 - $C_3 = 46468979279750354732637631044,$
 - $C_4 = 32870167545903741339819671379$
- the text shall be deciphered!

The ciphertext is not
necessary for the actual
cryptanalysis
(locating the private key) !

Solution using CrypTool (more detailed in online help examples section):

- enter public parameters into “RSA cryptosystem” (menu Individ. Procedures)
- button “factorise the RSA modulus” yields prime factors $p, q = N$
- based on that information private exponent $d = e^{-1} \bmod (p-1)(q-1)$ is determined
- decrypt the cipher text with d : $M_i = C_i^d \bmod N$

The attack with CrypTool is workable for RSA moduli up to 250 bit

Then you could digitally sign for someone else !

Examples (III)

Short RSA modulus: enter public RSA parameters

The RSA Cryptosystem

RSA using the private and public key -- or using only the public key

☐ Choose two prime numbers p and q . The number $N = pq$ is the public RSA modulus and $\phi(N) = (p-1)(q-1)$ is the Euler number. Public key e is coprime to $\phi(N)$. The private key $d = e^{-1} \pmod{\phi(N)}$ is calculated from this.

☒ For the purpose of data encryption or certificate checking it is sufficient to enter the public RSA parameters: the RSA modulus N and the public key e .

Factorisation attack

You may try to factorise the public RSA modulus N into its primes p and q

Factorise RSA modulus...

RSA parameters

RSA modulus N	63978486879527143858831415041	(public)
$\phi(N) = (p-1)(q-1)$		(secret)
Public key e	17579	
Private key d		

Update parameters

RSA encryption using e / decryption using d

1. enter RSA parameters* N and e

2. factorise

* currently limited to numbers up to 2^{1024}

Examples (III)

Short RSA modulus: factorise RSA modulus

The screenshot shows the 'Factorisation of a Number' window in the Cryptool application. The 'Algorithms for factorisation' section on the left has several checkboxes, all of which are checked: Brute-force method, Brent algorithm, Pollard method, Williams method, Lenstra algorithm, and Quadratic sieve method. The 'Input' section on the right contains a text box with the number '63978486879527143858831415041'. Below the input section, the 'Factorisation' section displays the result: 'The factorisation is represented in the format $\langle z_1^{a_1} * z_2^{a_2} * \dots * z_n^{a_n} \rangle$. Composite numbers are appearing in red color.' The 'Last factorisation through:' dropdown menu is set to 'Pollard'. The 'Time required:' is '0,981 seconds.' The 'Factorisation result:' text box shows the result: '145295143558111 * 440334654777631'. A blue circle highlights this result, and a blue arrow points from it to a text box that says '3. factorisation yields p and q'. Another blue arrow points from this text box to the 'OK' button in a smaller 'CrypTool' dialog box that is overlaid on the main window. The dialog box contains an information icon and the text: 'The RSA modulus N has been successfully factorised into the primes p and q! You can now perform the RSA operation with the secret key d. For this purpose just click the button Decrypt.' The 'OK' button is highlighted with a dashed border. The main window also has 'Details' and 'Close' buttons at the bottom right.

Factorisation of a Number

Algorithms for factorisation

- ☒ Brute-force method
- ☒ Brent algorithm
- ☒ Pollard method
- ☒ Williams method
- ☒ Lenstra algorithm
- ☒ Quadratic sieve method

Input

Enter the number to be factorised:

63978486879527143858831415041

Factorisation

The factorisation is represented in the format $\langle z_1^{a_1} * z_2^{a_2} * \dots * z_n^{a_n} \rangle$. Composite numbers are appearing in red color.

Last factorisation through: Pollard

Time required: 0,981 seconds.

Factorisation result:

145295143558111 * 440334654777631

3. factorisation yields p and q

CrypTool

The RSA modulus N has been successfully factorised into the primes p and q! You can now perform the RSA operation with the secret key d. For this purpose just click the button Decrypt.

OK

Details

Close

Examples (III)

Short RSA modulus: determine private key d

The RSA Cryptosystem

☐ RSA using the private and public key -- or using only the public key

☒ Choose two prime numbers p and q. The number $N = pq$ is the public RSA modulus and $\phi(N) = (p-1)(q-1)$ is the Euler number. Public key e is coprime to $\phi(N)$. The private key $d = e^{-1} \pmod{\phi(N)}$ is calculated from this.

☐ For the purpose of data encryption or certificate checking it will do with the published RSA parameter: the RSA modulus N and the public key e.

Prime number entry

Prime number p: 145295143558111

Prime number q: 440334654777631

Generate prime numbers

RSA parameters

RSA modulus N: 63978486879527143858831415041 (public)

$\phi(N) = (p-1)(q-1)$: 63978486879526558229033079300 (secret)

Public key e: 17579

Private key d: 10663687727232084624328285019

Update parameters

RSA encryption using e / decryption using d

Input as: ☒ text ☐ numbers

Options for alphabet and number system...

Enter either as text or as numbers in the format: number[1] # ... # number[n] (numbers of base 1240752)

Change the view to the owner of the secret key.

4. p and q have been entered automatically and secret key d has been calculated

5. adjust options

Examples (III)

Short RSA modulus: adjust options

Options for RSA Encryption

Text options

☐ All 256 ASCII characters

☒ Specify alphabet: Number of characters: 27

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Block length

The number of characters that are encrypted with each RSA operation.
The maximum size is subject to the length of the RSA modul N in bit, the number of characters in the alphabet and the method used for the coding.

Block length in characters: (Maximum block length 2 characters)

Number system

The numbers for RSA encryption and decryption will be represented in the following number system

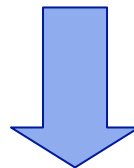
☒ Decimal ☐ Binary ☐ Octal ☐ Hexadecimal

OK Cancel

6. select alphabet

7. select coding method

8. select block length



Examples (III)

Short RSA modulus: decrypt cipher text

RSA parameters

RSA modulus N	63978486879527143858831415041	(public)
$\phi(N) = (p-1)(q-1)$	63978486879526558229033079300	(secret)
Public key e	17579	
Private key d	10663687727232084624328285019	

[Update parameters](#)

RSA encryption using e / decryption using d

Input as ☐ text ☒ numbers [Options for alphabet and number system...](#)

Ciphertext coded in numbers of base 10

7091621432020076311552201 # 46468979279750354732637631044 # 32870167545903741339819671279

Decryption into plaintext $m[i] = c[i]^d \pmod{N}$

00000000000001401202118011200 # 00000000000001421130205181900 # 0000000000000011805001301

Output text from the decryption (into segments of size 8; the symbol '#' is used as separator).

NATURAL # NUMBERS # ARE MADE # BY GOD

Plaintext

NATURAL NUMBERS ARE MADE BY GOD

[Encrypt](#) [Decrypt](#) [Close](#)

9. enter cipher text

10. decrypt

Examples (IV)

Analysis of encryption used in the PSION 5

Practical application of cryptanalysis:

Attack on the encryption option in the PSION 5 PDA word processing application

Starting point: an encrypted file on the PSION

Requirements

- encrypted English or German text
- depending on method and key length, 100 bytes up to several kB of text

Procedure

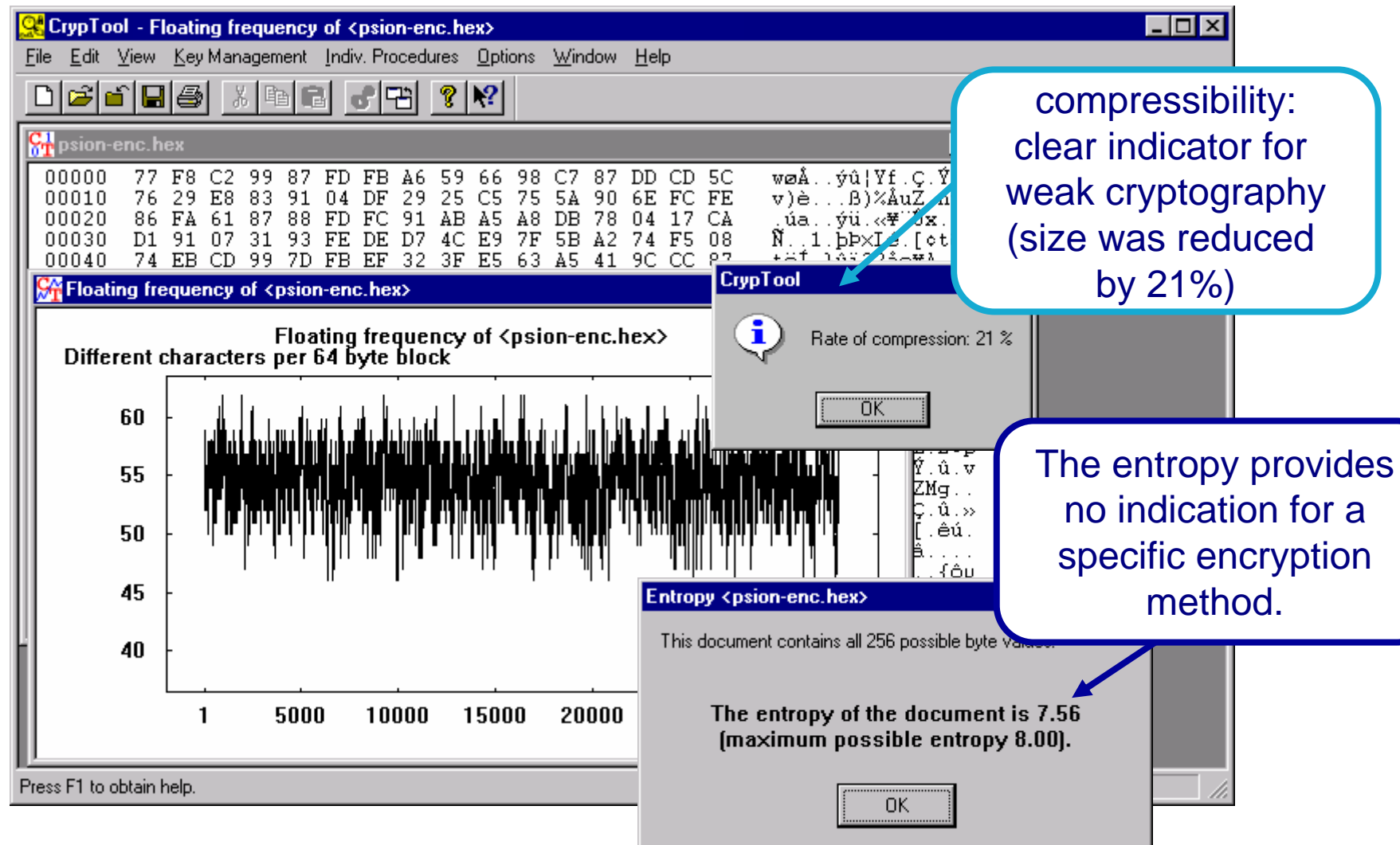
- pre-analysis
 - entropy
 - floating entropy
 - compression test
- auto-correlation
- try out automatic analysis with classical methods

} ⇒ **probably classical
encryption algorithm**



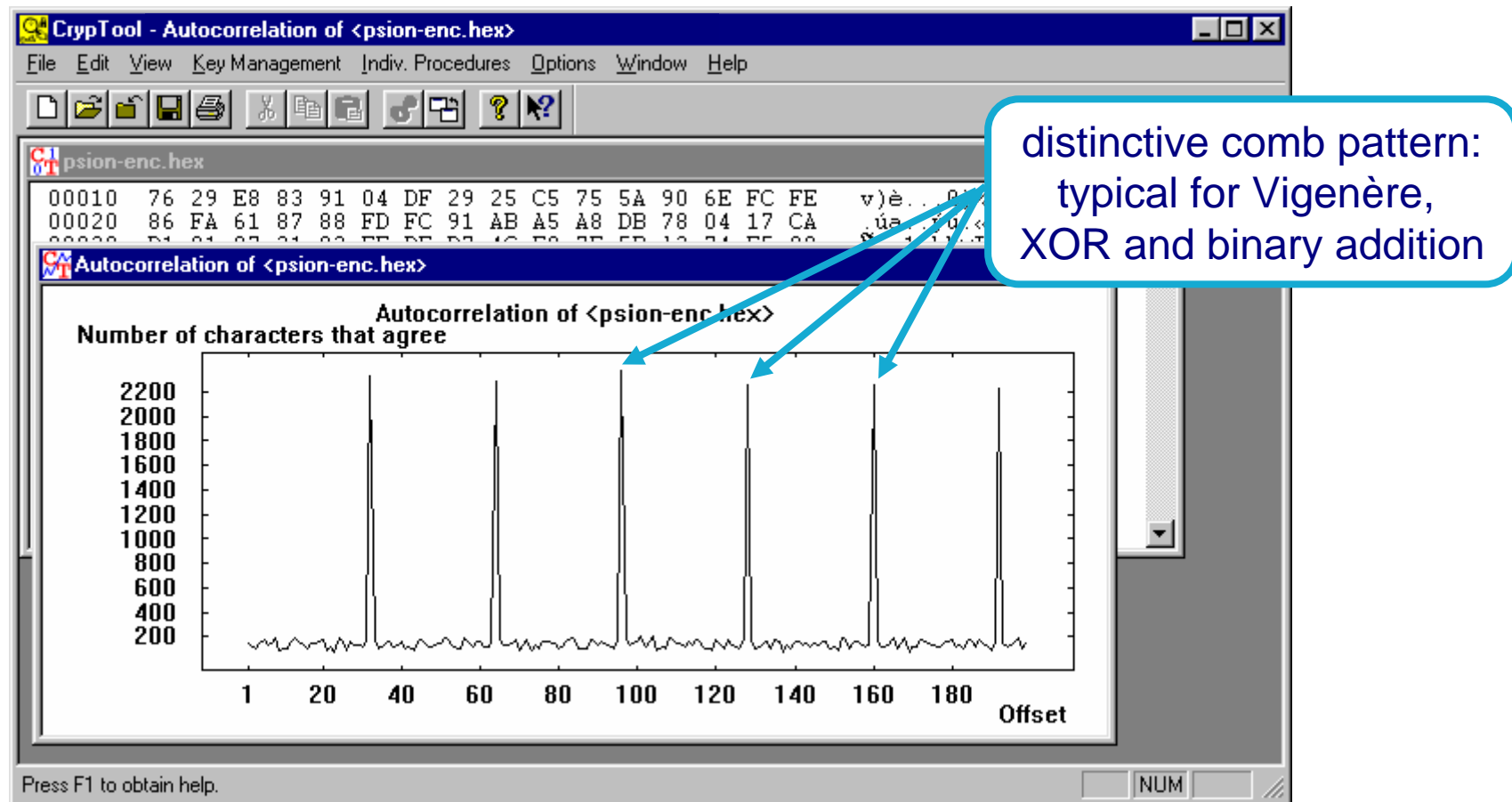
Examples (IV)

PSION 5 PDA – determine entropy, compression test



Examples (IV)

PSION 5 PDA – determine auto-correlation



* The encrypted file is available with CrypTool (see CrypTool\examples\psion-enc.hex)

Examples (IV)

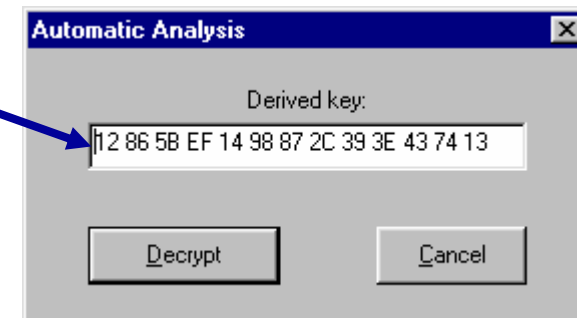
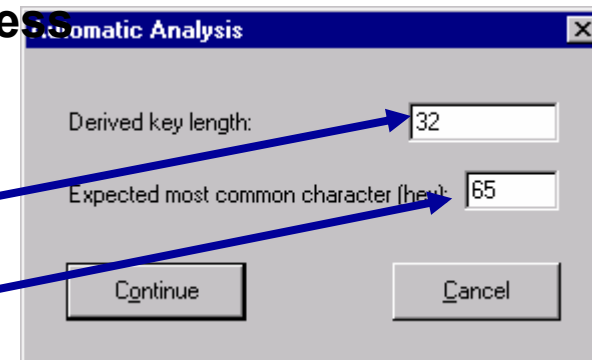
PSION 5 PDA – automatic analysis

Automatic analysis using Vigenère: no success

Automatic analysis using XOR: no success

Automatic analysis using binary addition:

- CrypTool calculates the key length using auto-correlation: 32 bytes
- The user can choose which character is expected to occur most frequently: “e” = 0x65 (ASCII code)
- Analysis calculates the most likely key (based on the assumptions about distribution)
- Results: good, but not perfect



Examples (IV)

PSION 5 PDA – results of automatic analysis

Results of automatic analysis with assumption “binary addition”:

- results good, but not perfect: 24 out of 32 key bytes correct.
- the key length 32 was correctly determined. ←

Address	Hex Data	Text Fragments
00000	65 72 67 AA 73 65 74 7A 20 28 55 53 74 47 29 06	erg³setz (UStG).
00010	06 06 8A 72 73 74 65 72 65 41 62 B8 A8 68 6E AE	...rstereAb,`hn@
00020	74 74 06 98 74 65 75 65 72 67 65 67 65 6E 73 74	tt...teuergegenst
00030	61 6E A9 20 75 6E 64 20 8C 65 6C B9 BA 6E 67 B8	an@ und .el¹ng,
00040	62 65 72 AA 69 63 68 06 06 A7 20 31 2E 06 28 31	ber³ich...\$ 1...(1
00050	29 20 89 65 72 20 55 6D B8 61 74 BF B8 74 65 BA) .er Um,at¿,te²
00060	65 72 20 BA 6E 74 65 72 6C 69 65 67 65 6E 20 64	er ²nterliegen d
00070	69 65 65 66 6F 6C 67 65 B3 64 65 B3 65 55 6D B8	ieefolge³de³eUm,
00080	E4 74 7A AA 3A 06 31 2E 20 64 69 65 20 4C 69 65	ätz³...l. die Lie
00090	66 65 B7 75 6E 67 65 6E 65 75 6E A9 65 73 6F B3	fe.ungeneun@eso³
000A0	73 74 69 AC 65 6E 20 4C 65 69 73 74 75 6E 67 65	stiren Leistunge
000B0	6E 2C 65 64 69 65 20 65 AE 6E 20 9A B3 74 65 B7	n,edie e@n .³te
000C0	6E 65 68 B2 65 72 20 69 6D 20 49 6E 6C 61 6E 64	neh²er im Inland
000D0	20 67 AA 67 65 6E 20 45 B3 74 67 AA B1 74 20 AE	g³gen E³tg³tt ®
000E0	6D 20 52 A6 68 6D 65 6E 20 73 65 69 6E 65 73 20	m R³hmen seines
000F0	55 6E B9 65 72 6E 65 68 B2 65 6E B8 65 61 75 B8	Un¹erneh²en,eau.

- the password entered was not 32 bytes long.
⇒ PSION Word derives the actual key from the password.
- manual post-processing produces the encrypted text (not shown)

Examples (IV)

PSION 5 PDA – determining the remaining key bytes

Copy key to clipboard during automatic analysis

In automatic analysis hex dump,

- determine incorrect byte positions, e.g. 0xAA at position 3
- guess and write down corresponding correct bytes: „e“ = 0x65

In encrypted initial file hex dump,

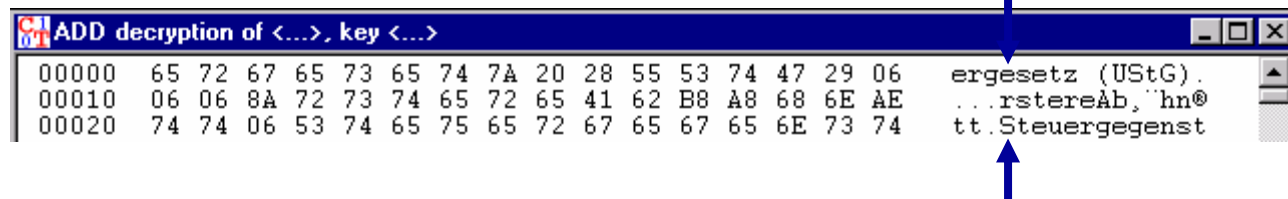
- determine initial bytes from the calculated byte positions: 0x99
- calculate correct key bytes with CALC.EXE: $0x99 - 0x65 = 0x34$

Correct key from the clipboard

- 12865B341498872C393E43741396A45670235E111E907AB7C0841...

Decrypt encrypted initial document using binary addition

- bytes at position 3, 3+32, 3+2*32, ... are now correct



Examples (V)

Weak DES key

The top screenshot shows the CrypTool 1.4.00 interface. The 'Crypt/Decrypt' menu is open, and 'DES (ECB)...' is selected. A 'Key Entry: DES (CBC)' dialog box is open, showing a key length of 64 bit and a key value of 01 01 01 01 01 01 01 01. A text box below the dialog box contains the text: - encrypt 2 times with... - results in plaintext.

The bottom screenshot shows the CrypTool 1.4.00 interface with the 'DES (ECB) encryption of <...>, key <01 01 01 01 01 01 01 01>' window open. The window displays the encryption of 'startingexample-en.txt' using the key 01 01 01 01 01 01 01 01. The resulting ciphertext is shown in hexadecimal and ASCII. The ASCII output is 'DES weak key demo...', which is the original plaintext.

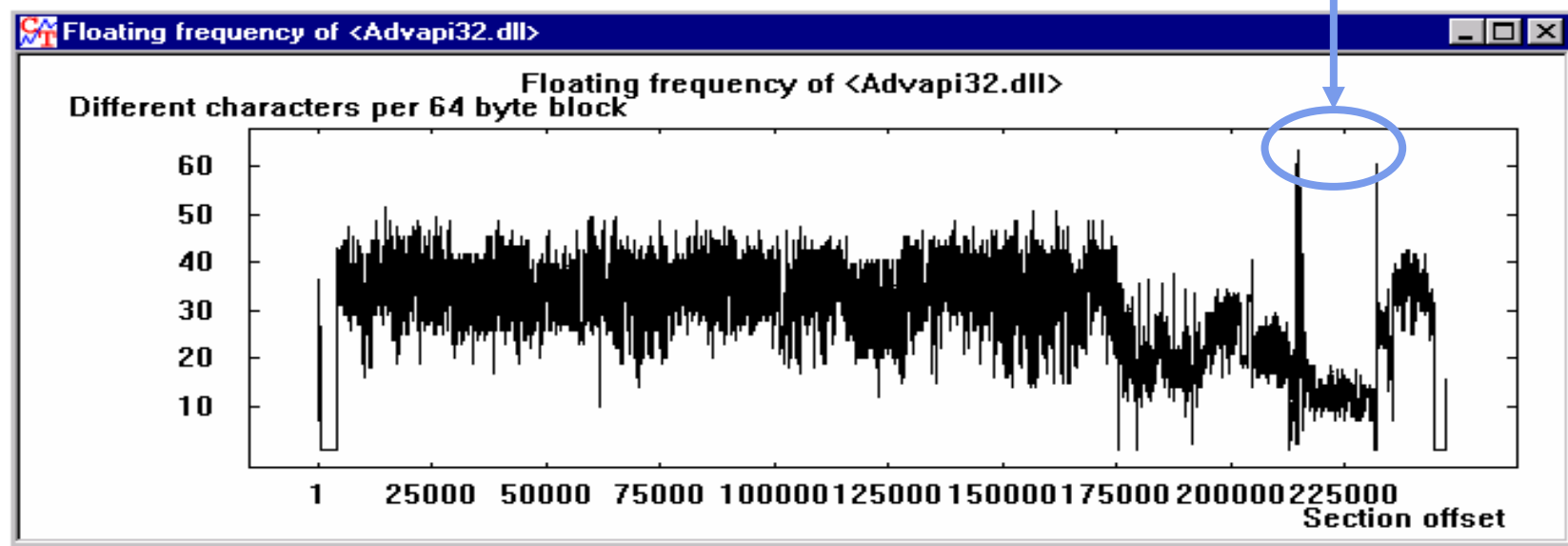
Examples (VI)

Locate key material

The function “Floating frequency” is suitable for locating key material and encrypted areas in files.

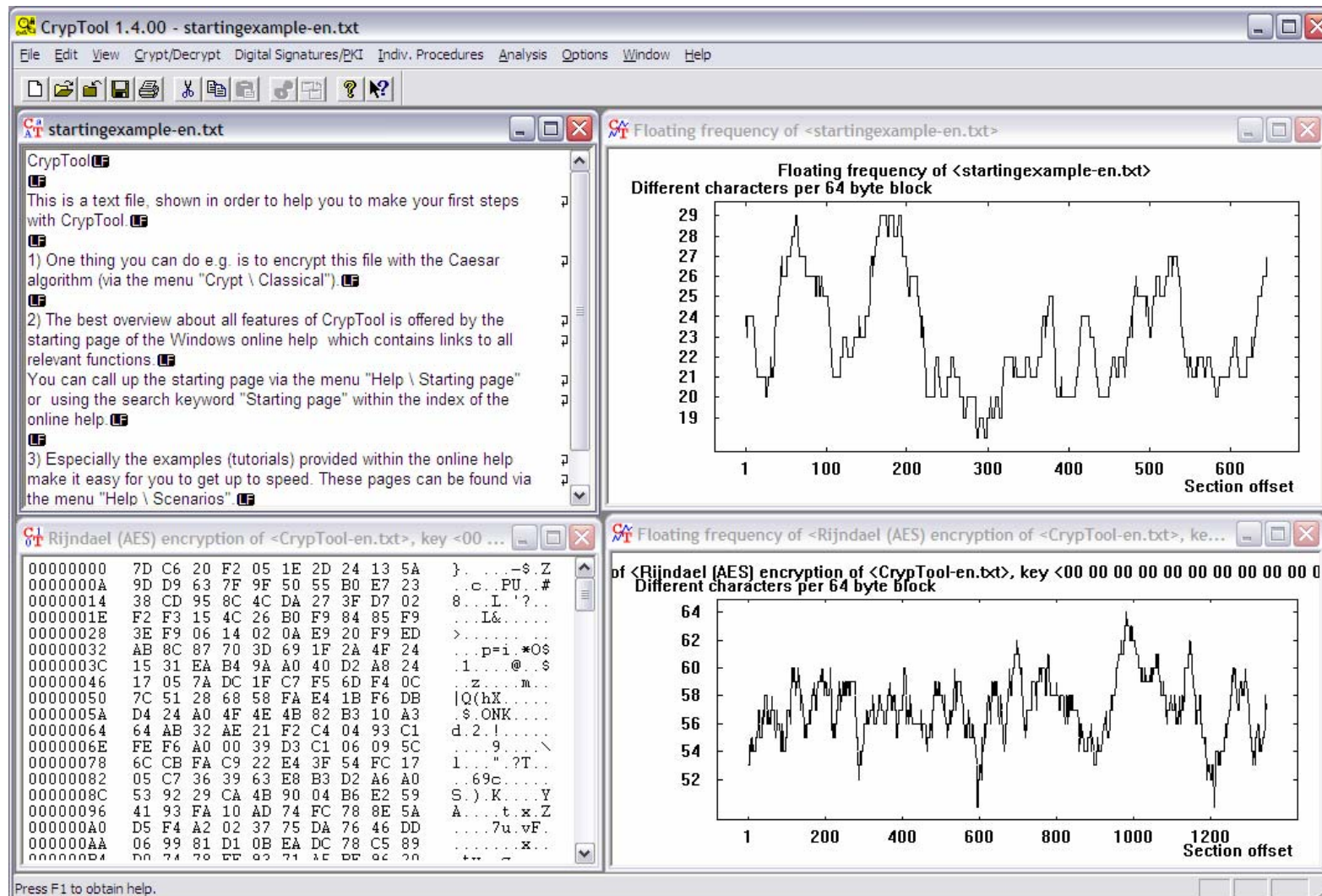
Background:

- key data is “more random” than text or program code
- can be recognised as peaks in the “floating frequency”
- example: the “NSAKEY” in advapi32.dll



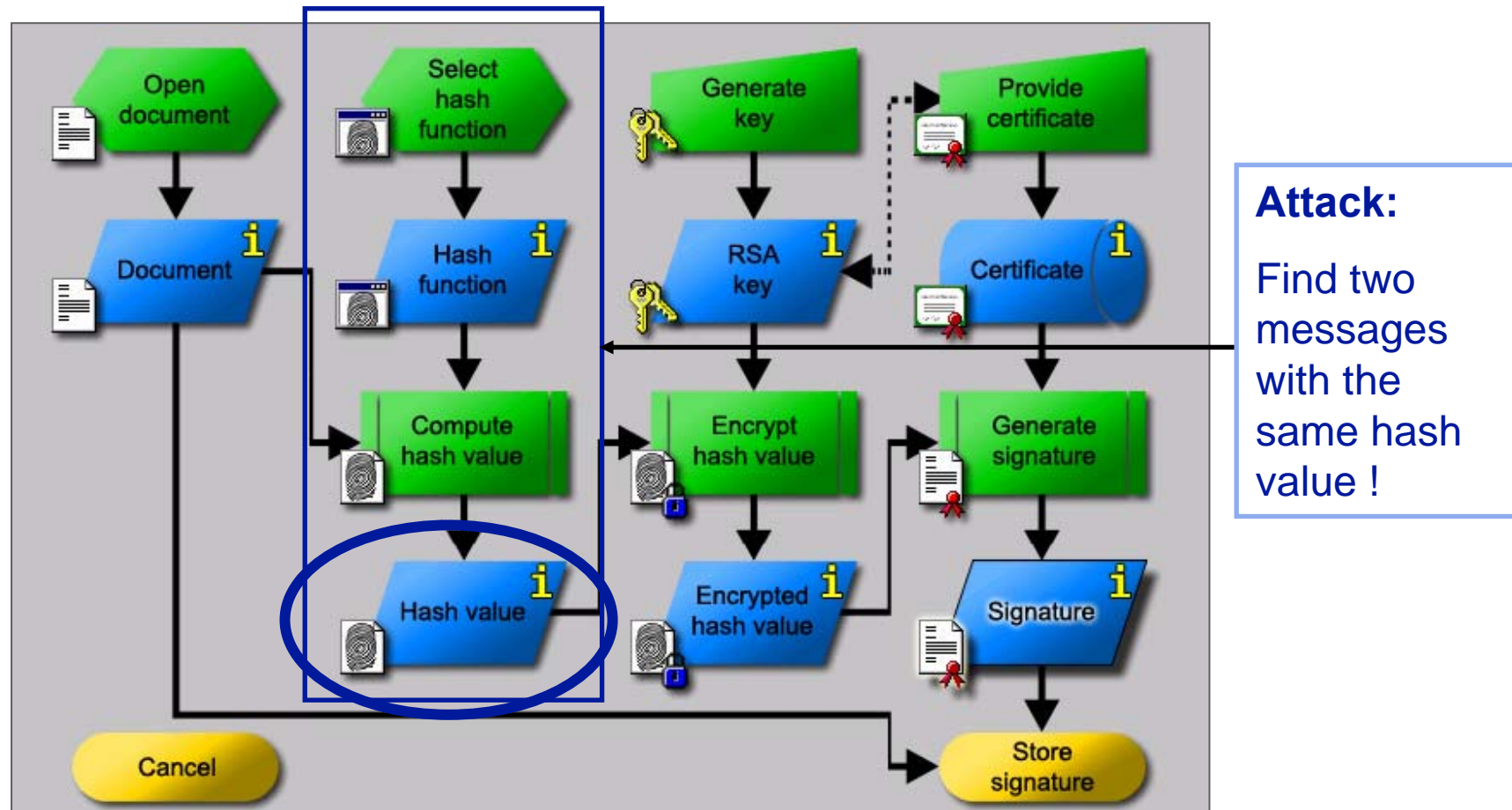
Examples (VI)

Comparison on floating frequency with other files



Examples (VII)

Attack on digital signature



Examples (VII)

Attack on digital signature – idea (I)

Attack on the digital signature of an ASCII text based on hash collision search

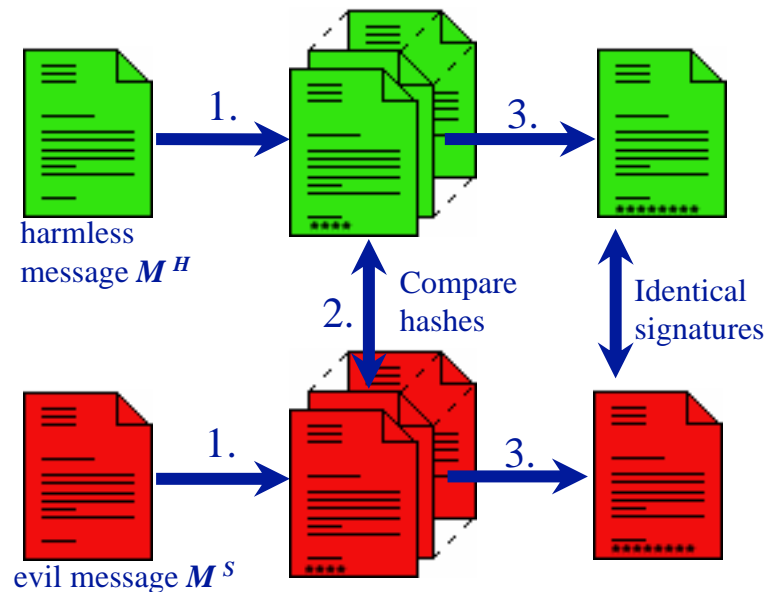
Idea:

- ASCII-Texts can be modified by changing/inserting ***non-printable*** characters, without changing the visible content
- modify two texts in parallel until a hash collision is found
- exploit the birthday paradox (birthday attack)
- generic attack applicable to all hash functions
- can be run in parallel on many machines (not implemented)
- implemented in CrypTool as part of his bachelor thesis “*Methods and tools for attacks on digital signatures*” (German), 2003.

Concepts: Mappings, modified Floyd algorithm (constant memory consumption) !

Examples (VII)

Attack on digital signature – idea (II)



1. **Modification:** starting from a message M create N different messages M_1, \dots, M_N with the same “content” as M .
2. **Search:** find modified messages M_i^H and M_j^S with the same hash value.
3. **Attack:** the signatures of those two documents M_i^H and M_j^S are the same.

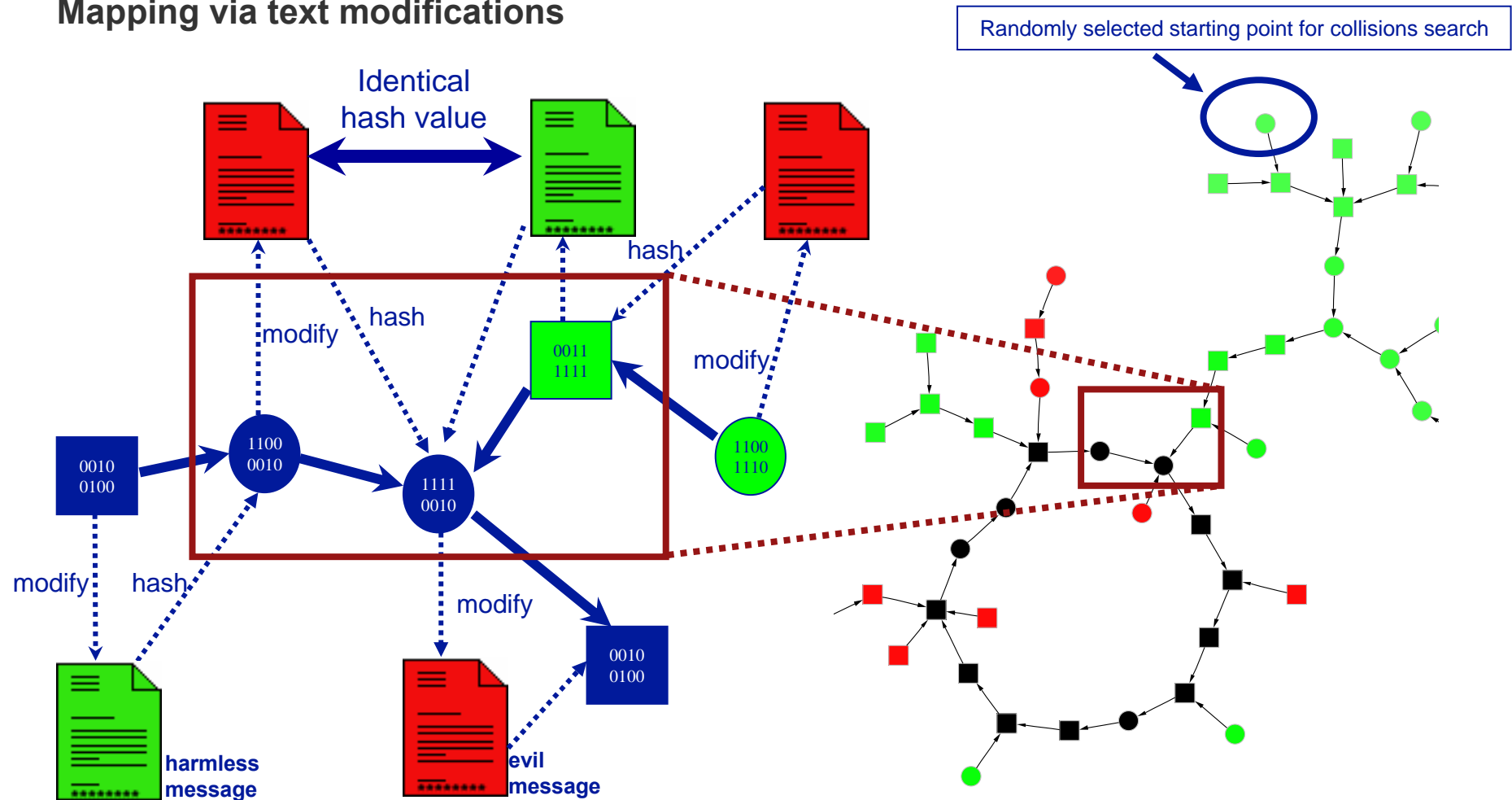
We know from the birthday paradox that for hash values of bit length n :

- search collision between M^H and M_1^S, \dots, M_N^S : $N \approx 2^n$
- search collision between M_1^H, \dots, M_N^H and M_1^S, \dots, M_N^S : $N \approx 2^{n/2}$

↑
Estimated number of generated messages
in order to find a hash collision.

Locate Hash Collisions





Mapping via text modifications

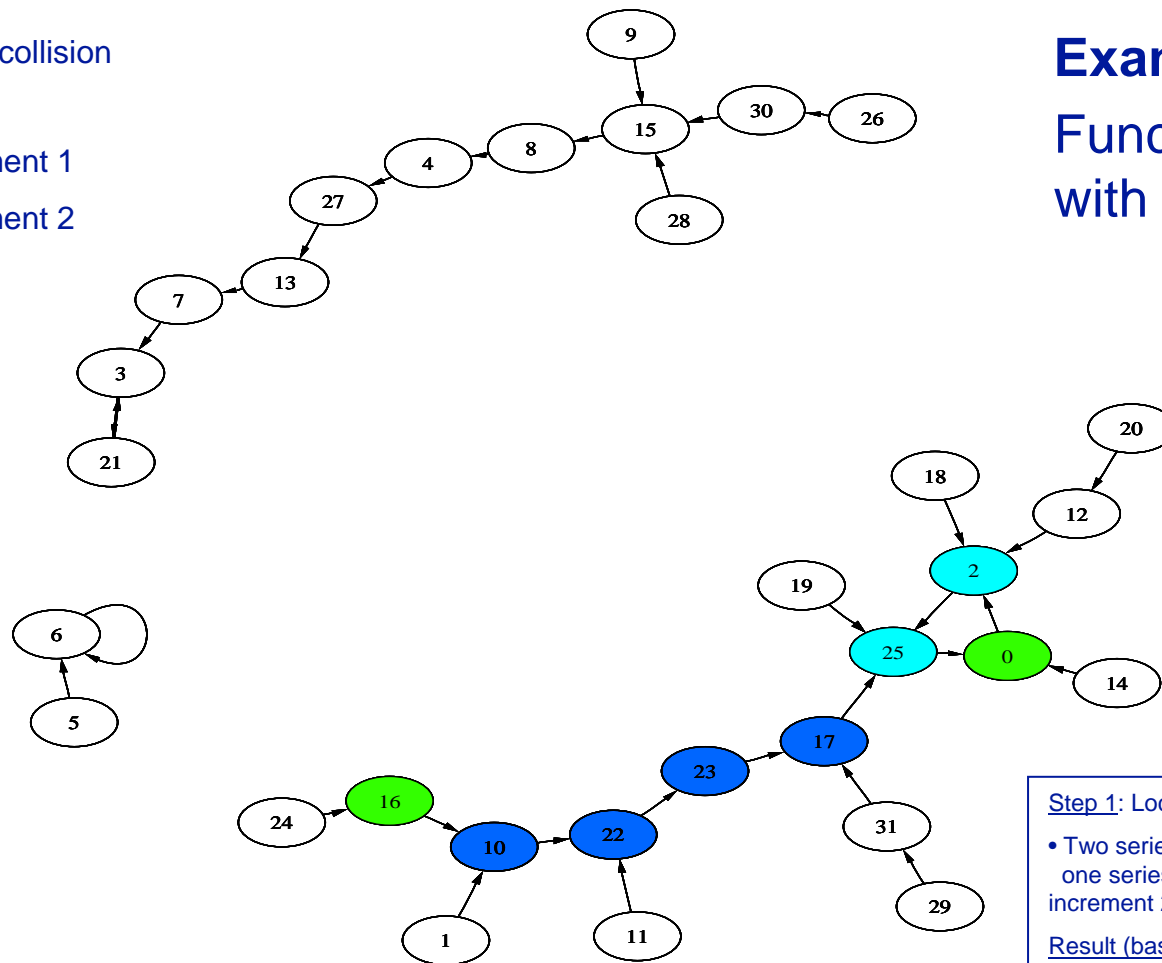


- green / red: path from a tree to the cycle – this can lead to a useful or useless collision.
- square / round: hash value has even / odd parity
- black: all nodes within the cycle

Locate Hash Collisions

Floyd-Algorithm: meet within the cycle

-  start / collision
-  cycle
-  increment 1
-  increment 2



Example:
Function graph
with 32 nodes

Step 1: Locate matching point within cycle:

- Two series with identical starting point [16]:
 - one series with increment 1, the other with increment 2.

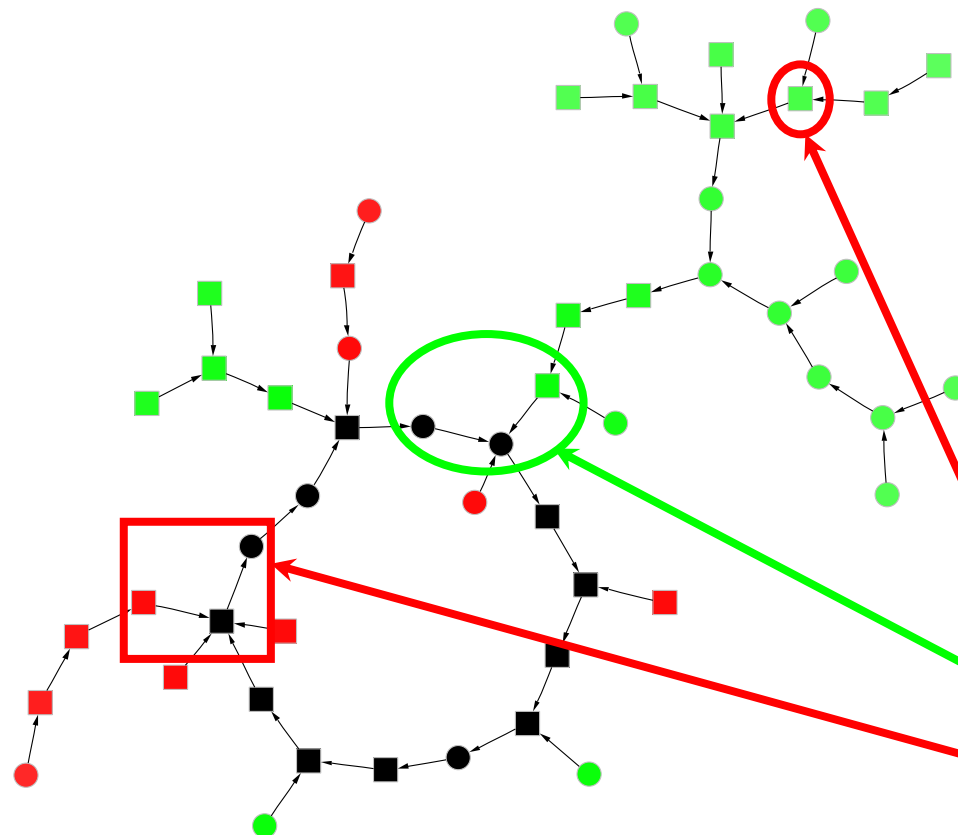
Result (based on graph theory):

- both series always end up in a cycle.
- both series match in a node within the cycle (in this case 0).

Step into cycle (Extension of Floyd): find entry point

- The predecessors (in this case 17 and 2) result in a hash collision.

Birthday paradox attack on digital signature



Real examination of Floyd algorithm

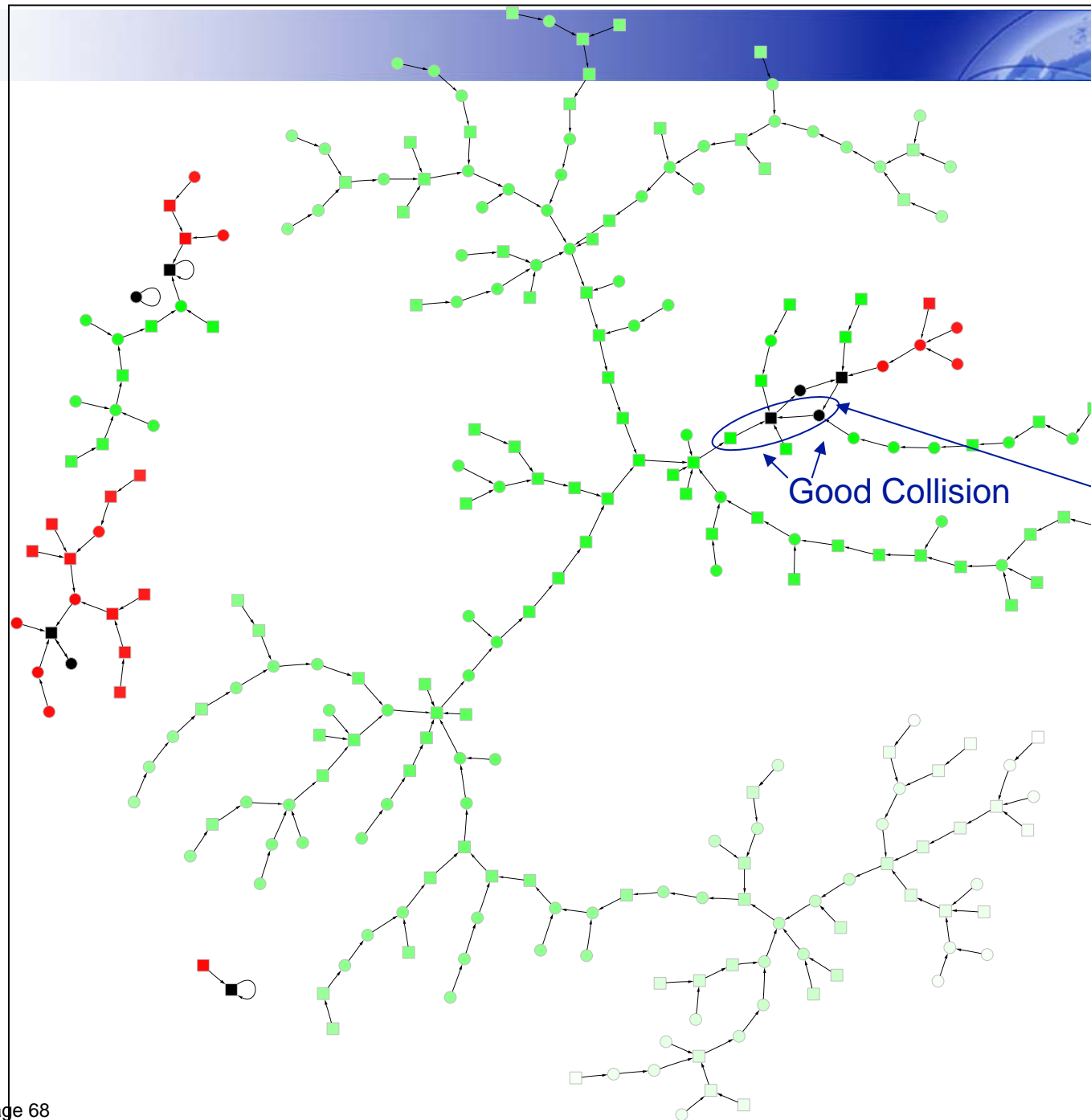
- Visual and interactive presentation of the Floyd algorithm („Moving through the mapping" into a cycle).
- Adaptation of the Floyd algorithm for a digital signature attack.

Starting point

Good collision

Bad collision

* The Floyd algorithm is implemented, but the visualization of the algorithm is not yet implemented in CrypTool.



An example for a
“good” Mapping
(nearly all nodes
are green).

In this graph
almost all nodes
belong to a big
tree, which leads
into the cycle with
an even hash
value and where
the entry point
predecessor
within the cycle is
odd.

That means that
the attacker finds
a useful collision
for nearly all
starting points.

Examples (VII)

Attack on digital signature: Attack

Attack on the Hash Value of the Digital Signature

In this implemented attack at hand the attacker tries to find two different messages which translate to the same hash value.

1. Choose "harmless" file

The attacker assumes that his victim will digitally sign the "harmless" message due to its non vicious content.

C:\Programme\CrypTool-1-3-04-beta-8-en\original.txt Browse ...

2. Choose "dangerous" file

In case of a successful attack the attacker can argue that his victim has digitally signed the "dangerous" instead of the "harmless" message.

C:\Programme\CrypTool-1-3-04-beta-8-en\fake.txt Browse ...

Start search / Set options

By clicking "Start search" you initiate the attack which searches for two modifications of the messages above which translate to the same hash value.

The semantics of the messages do not vary throughout the attack as only unprintable or formatting characters are used to modify them.

In the "Options" you can set the hash function, the number of bits of the hash values that have to match and the method of modifying the messages.

3. Options ...

4. Start search Cancel

Options for the attack on the hash value of the digital signature

Hash function

Choose one of the six hash functions and set the number of bits that have to match between two hash values so that the attack is considered successful.

☐ MD2 ☐ MD4 ☒ MD5

☐ SHA ☐ SHA-1 ☐ RIPEMD-160

Significant bit length 32 Co-domain: 1 - 128

Options for the modification of messages

Determine the way messages are modified

☐ Insert blanks ☒

☒ Attach characters ☐

Apply Reset to standard

Searching for a pair of messages ...

Run 1
Cycle search
Progress: 36% remaining time: 00:00:29

Searching for a pair of messages ...

Run 1
Collision search
Progress: 53% remaining time: 00:00:21

Cancel

Examples (VII)

Attack on digital signature: Results

Harmless message: MD5, <A9 76 34 AB>

Dear Mr Shopaholic,
please order a typewriter.
Regards
Honest John

Dangerous message: MD5, <A9 76 34 AB>

Dear Mr Shopaholic,
please order a Porsche and a prepaid insurance scheme for Mr. Dodge.
Regards
Honest John

MD5: A9 76 34 AB
65 53 37 21 6F 70
1F 6C 6C 3F 2F 6D

MD5: A9 76 34 AB
AC 10 96 30 CC 47
24 D5 70 D8 84 71

The first 32 bits of the hash values are identical.

Experimental results

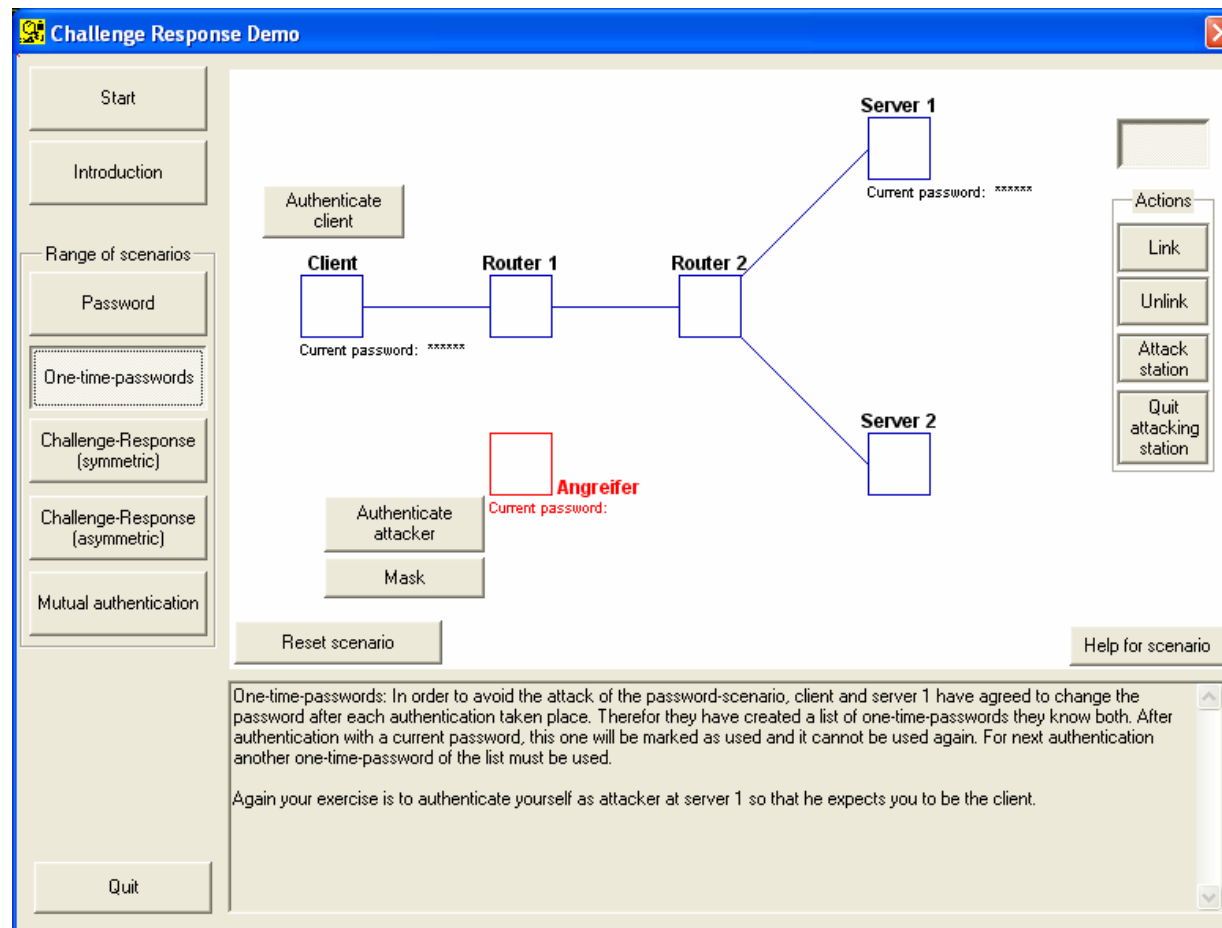
- 72 Bit *partial collision* (equality of the first 72 hash value bits) were found in a couple of days on a single PC.
- Signatures using hash values of up to 128 bit can be attacked today using massive parallel search!

In addition to the interactive handling:

Automated offline feature in CrypTool: Execute and log the results for entire sets of parameter configurations. Available through command line execution of CrypTool.

Examples (VIII)

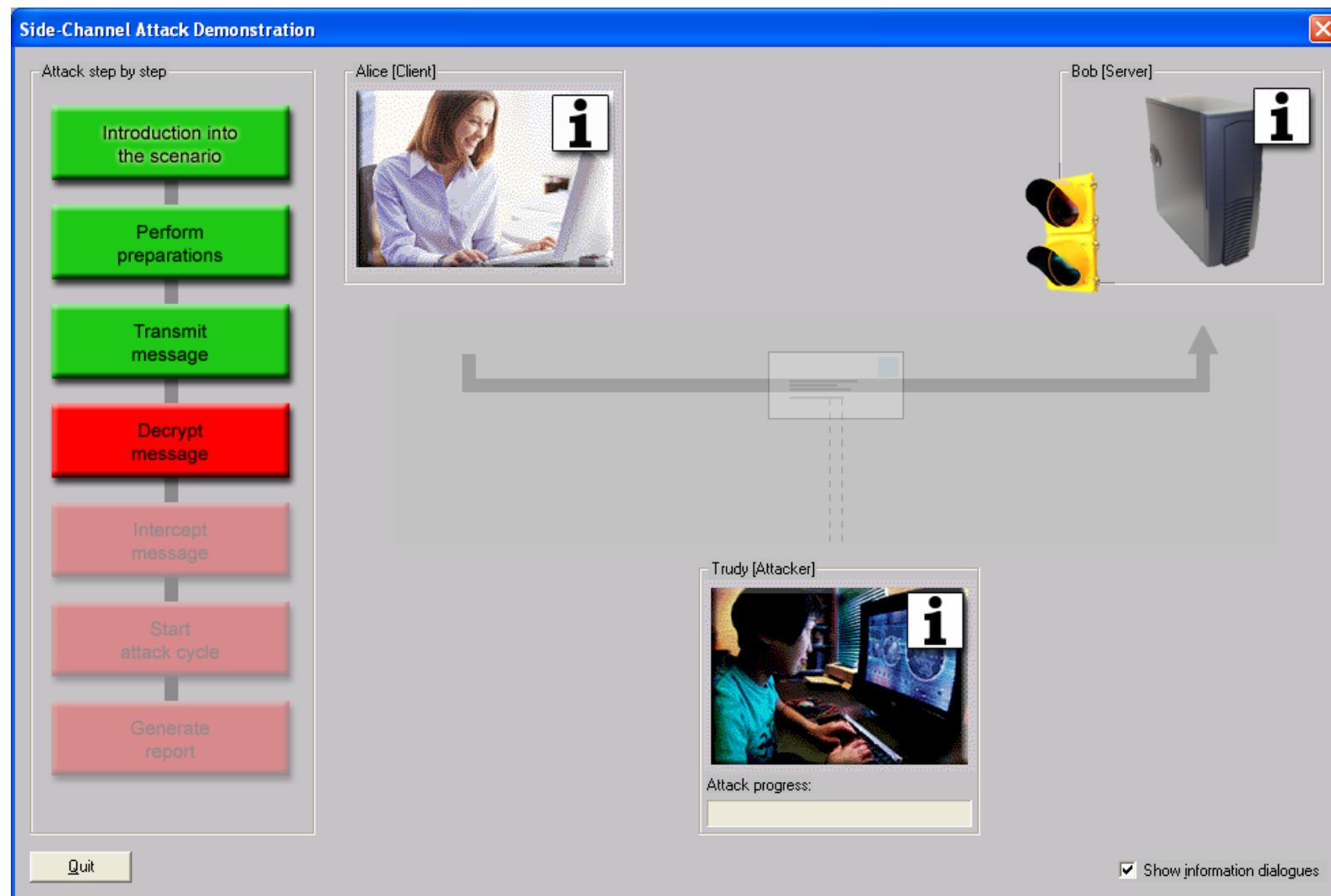
Authentication in a client server environment



- Interactive demo for different authentication methods.
- Defined opportunities of the attacker.
- You can play the role of an attacker.
- **Learning effect:** Only two-sided authentication is secure.

Examples (IX)

Demonstration of a side-channel attack (at a hybrid encryption protocol)



Examples (IX)

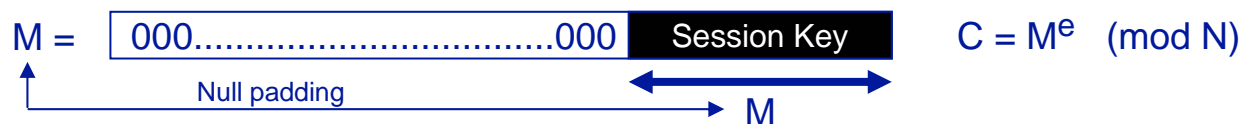
Idea for this side channel attack

- Ulrich Kühn, *Side-channel attacks on textbook RSA and ElGamal encryption* (2003)

Prerequisites:

- RSA encryption: $C = M^e \pmod{N}$ and decryption: $M = C^d \pmod{N}$.
- 128-Bit session keys (in M) are „word book encoded“ (Null padding).
- The server knows the secret key d and
 - uses after decryption the 128 least significant bits only (no validation of Zero padding bits) (that means the server does not recognize if there is something other than zero).
 - Prompts an error message, if the encryption attempt results in a wrong session key (decrypted text can not be interpreted by the server). In all other cases there will be no message.

Idea for attack: approximation for Z out of the equation $N = M * Z$ via $M = \lfloor N/Z \rfloor$



All bit positions for Z are successively calculated: for every step one gets 1 bit more. The attacker modifies C to C' (see below). If a bit overflow occurs while calculating M' on the server (recipient), the server sends an error message. Based on this information the attacker gets a bit for Z.



Example (X)

Mathematics: Attacks on RSA using lattice reduction

Attack on small secret exponents (according to Bloemer / May)

Description
This attack allows to factor an RSA modulus N , in case the secret key d is chosen too small compared to N . The number $\delta = \log(d)/\log(N)$ is called "size of d ". The attack is feasible for $\delta < 0.290$.
☐ In order to apply examples from the literature, first enter the public key (N, e) . Afterwards enter the estimated value of δ . Alternatively you can enter d directly which is used to calculate δ .
☒ In order to generate a random example enter the desired parameters δ and bitlength of N . By clicking "Generate random key" the keys are generated.
Then click "Start".

Step 1: Enter key parameters and key

Bitlength of N : δ :

N :

e :

d :

Step 2: Enter attack parameters for the lattice base reduction

m : Determines the size of the lattice to reduce and the maximal size of δ . Should be at least 4.

t : Is optimally calculated as a function of m .

Lattice dimension: Size of the lattice to reduce. Has major impact on the runtime.

Maximal δ : Maximal size of δ for big N ($N > 1000$ Bit).

Step 3: Start attack

Building lattice:

Reducing lattice: Reductions:

Calculating resultant: Resultants:

Overall time:

Found factorization:
 p : q :

- Shows how the parameters of the RSA method have to be chosen, so that the algorithm resists the lattice reduction attacks described in current literature.
- **3 variants**
 1. The secret exponent d is too small in comparison to N .
 2. One of the factors of N is partially known.
 3. A part of the plaintext is known.
 - These assumptions are realistic.
- The current estimation is displayed.

Examples (X)

Example page of the online help

Help for CrypTool 1.4.00 beta 10

Ausblenden Zurück Vorwärts Abbrechen Aktualisieren Startseite Drucken Optionen

Inhalt Index Suchen

- Start with CrypTool
 - Starting page: An Introduction
 - How to work through the examples
 - How to use the CrypTool only
- Functionality of CrypTool
- Work on a document
- Change CrypTool options
- CrypTool Menus
 - Menu File
 - Menu Edit
 - Menu View
 - Menu Crypt/Decrypt
 - Menu Digital Signatures/PKI
 - Menu Individual Procedures
 - Menu Analysis
 - Menu Options
 - Menu Window
 - Menu Help

Commands of the menu Lattice Based Attacks on RSA (Menu [Individual Procedures](#) \ RSA Cryptosystem)

The menu **Lattice Based Attacks on RSA** contains the following commands:

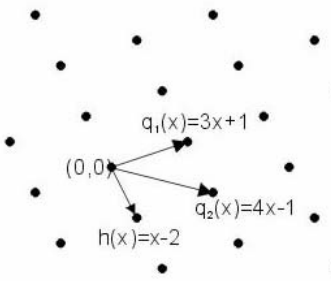
Factoring with a Hint	Attacks RSA with lattice reduction algorithms, if a part of one of the primes of N is known.
Attack on Stereotyped Messages	Attacks RSA with lattice reduction algorithms, if a part of the original cleartext of an intercepted ciphertext is known and if e is small.
Attack on Small Secret Keys	Attacks RSA with lattice reduction algorithms, if d is too small compared to N.

All attacks presented here are based on a common approach: first the task of breaking RSA is transformed into finding the root of a polynomial modulo an integer (mostly N) but to find such a root is a difficult problem.

To solve this problem further polynomials are generated which are known to have the same root. From the coefficients of these polynomials a latticebase is built. This is then reduced with, i.e. the LLL-algorithm to find a small vector.

From this newly found short vector a new polynomial is built. It can be proven that if the vector is short enough, the polynomial has the desired root not only modulo N, but also over the integers.

Example:



The polynomial $q_1(x) = 3x+1$ has a root x_0 modulo 7. It is supposed, that the polynomial $q_2(x) = 4x-1$ has the same root x_0 modulo 7. From these polynomials the vectors $b_1=[3 \ 1]$ and $b_2=[4 \ -1]$ are built. All integer linear combinations of these vectors form points in a lattice. The Figure on the left shows a part of this lattice. Each point of the lattice now can again be interpreted as a polynomial having the desired root. A short vector of the lattice is $b_3=[1 \ -2]$ from which the polynomial $h(x) = x-2$ is built. This polynomial has a root in $x_0=2$ over the integers as well as modulo 7. That $x_0=2$ is also a root of the polynomials $q_1(x)$ and $q_2(x)$ modulo 7 can be easily established.
($3x_0+1=7, 7 \bmod 7 = 0$)

Annotations:

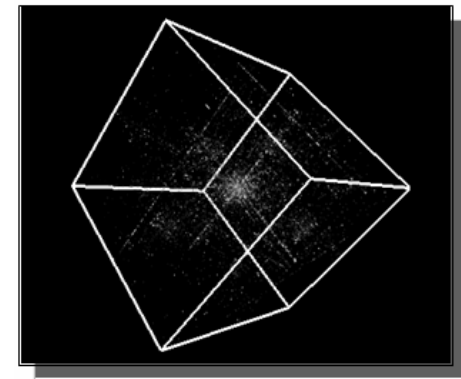
Examples (XI)

Random analysis with 3-D visualisation

3-D visualisation for random analysis

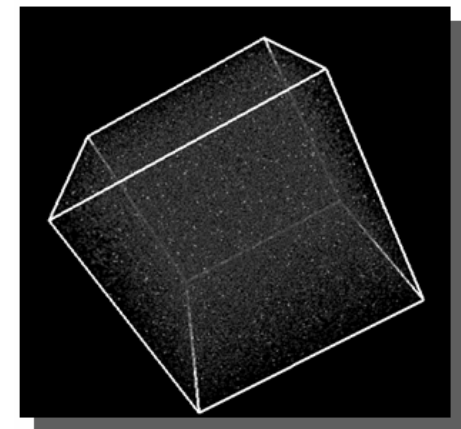
Example 1

- Open an arbitrary file (e.g. report in Word or PowerPoint presentation)
- It is recommended to select a file with at least 100 kb
- 3-D analysis using „Analysis“ \ „Analyse Randomness“ \ „3-D Visualization...“
- Result: **structures are recognisable**



Example 2

- Generation of random numbers („Indiv. Procedures“ \ „Generate Random Numbers...“)
- It is recommended to generate at least 100.000 random bytes
- 3-D analysis using „Analysis“ \ „Analyse Randomness“ / „3-D Visualization...“
- Result: uniform distribution (**no structures are recognisable**)

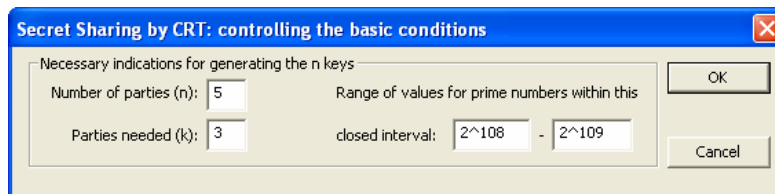


Examples (XII)

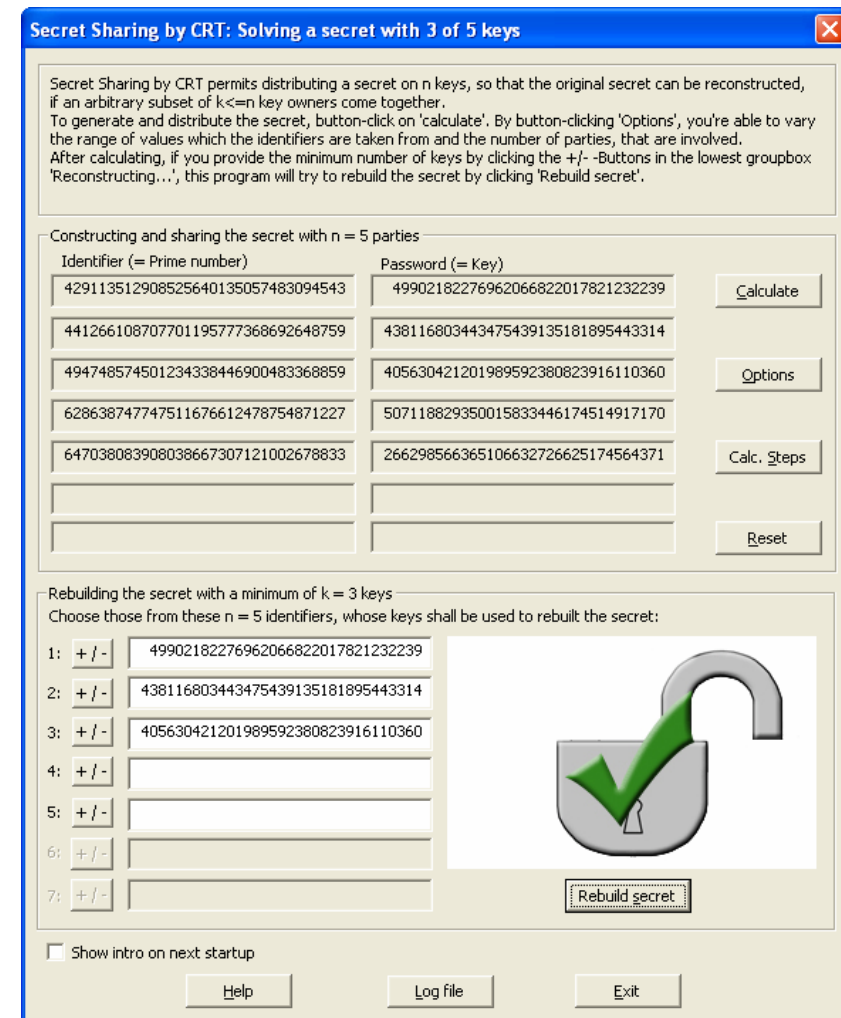
Secret Sharing with CRT – Implementation of the Chinese Remainder Theorem

Secret Sharing Example (I):

- Problem:
 - 5 people get a single key
 - To gain access at least 3 of the 5 people have to be present
- **CrypTool**: Menu „Indiv. Procedures“ \ „Chinese Remainder Theorem Applications“ \ „Secret Sharing by CRT“
- „Options“ allows to configure more details of the method.



- „Calc. Steps“ shows all steps to generate the key.



Examples (XII)

Shamir Secret Sharing

Secret Sharing Example (II):

- Problem
 - A secret value should be split for n people.
 - t out of n people are required to restore the secret value K .
 - (t, n) threshold scheme
- **CrypTool**: Menu „Indiv. Procedures“ \ „Secret Sharing Demo...“
 1. Enter the secret K , number of persons n and threshold t
 2. Generate polynomial
 3. Use parameters
- Using „**Reconstruction**“ the secret can be restored

Secret Sharing: Initializing the threshold scheme

By means of a (t, n) Shamir scheme a secret S can be distributed among n persons. Afterwards, t persons ($t \leq n$) will be able to reconstruct the original secret by combining their individual secrets (shares). To set up such a scheme, a polynomial $f(x)$ of degree at most $t-1$ (with $t-1$ coefficients $a[i]$ chosen at random) and a random prime p have to be generated. Each participant receives a randomly chosen public value x and his share, the corresponding secret value $y=f(x)$. For further details please check the CrypTool Online-Help by pressing F1.

Choose your secret and parameters to set up a scheme (whole numbers)

Secret S with $S >= 0$

Number of participants n with $n > 0$

Threshold (minimum) t with $t > 0$

Parameters concerning the polynomial $f(x)$ of degree $t-1$

All computations take place in the discrete space $GF(p)$

Polynomial $f(x)$

Prime p

Participants' values, calculated from chosen parameters:

participant	public value x	share (secret value $f(x)$)
<input checked="" type="checkbox"/> participant 1	6335	283
<input type="checkbox"/> participant 2	7059	1925
<input type="checkbox"/> participant 3	9449	898
<input type="checkbox"/> participant 4	6004	7490
<input checked="" type="checkbox"/> participant 5	1758	2557
<input type="checkbox"/> participant 6	196	8421
<input checked="" type="checkbox"/> participant 7	7521	6120
<input type="checkbox"/> participant 8	5023	5287

Please do select those participants which are to reconstruct the secret. To select several entries at a time, keep the Ctrl-Button pressed and use your mouse to mark your choice.

☐ Show information dialogue at startup

Example (XIII)

Implementation of CRT in astronomy to solve linear modular equation systems

Scenario in astronomy

- How long does it take until a given number of planets (with different rotation times) to become aligned.
- The result is a linear modular equation system, that can be solved with the Chinese remainder theorem (CRT).
- In this demo you can enter up to 9 equations and compute a solution using the CRT.

Example of use: Visualization of the Chinese Remainder Theorem in Astronomy - Planetary motion

By using the Chinese Remainder Theorem (CRT) you are able to solve linear modular equation systems. You can enter up to 9 equations $x = a[i] \bmod m[i]$ ($i=1, \dots, 9$) below and compute a solution. One can use such equation systems to determine the number of days until certain planets become aligned.

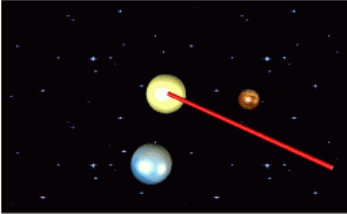
Simultaneous congruences/ modular linear equation systems

x ≡	15	mod	88
x ≡		mod	
x ≡	100	mod	365
x ≡		mod	
x ≡	0	mod	4327
x ≡		mod	
x ≡		mod	
x ≡	0	mod	60149
x ≡		mod	

Solution

126.228.390.655

Example of use in astronomy/ visualization fix



The period of the planets mercury and earth around the sun is 88 and 365 days. Up to reaching a certain radius vector s (red), it takes

15 and 100 days.

Can it occur, that mercury and earth are sometime once on the ray s ?

Choose a planet

<input checked="" type="checkbox"/> Mercury	<input type="checkbox"/> Mars	<input type="checkbox"/> Uranus
<input type="checkbox"/> Venus	<input checked="" type="checkbox"/> Jupiter	<input checked="" type="checkbox"/> Neptune
<input checked="" type="checkbox"/> Earth	<input type="checkbox"/> Saturn	<input type="checkbox"/> Pluto

In which time interval (days) does this incident repeat itself?

8.359.702.902.760

Examples (XIV)

Visualisation of symmetric encryption methods using ANIMAL (1)

Animated visualisation of several symmetric algorithms

- Caesar
- Vigenère
- Nihilist
- DES

CrypTool

- Menu „Indiv. Procedures“ \ „Visualization of algorithms using ANIMAL“ \ ...
- Interactive animation control using integrated control

The screenshot shows the 'Animal Animation: Caesar Cipher' window. At the top, there are two sliders: 'Animation speed' (0 to 1000) and 'Scaling of visualisation' (0 to 500). The main content area is titled 'Caesar-Cipher' and explains that it is a monoalphabetic substitution cipher using key k=3. It displays the plaintext 'GALLIA EST OMNIS DIVISA ...', the plaintext alphabet 'A B C D E F G H I J K L M N O P Q R S T U V W X Y Z', the ciphertext alphabet 'D E F G H I J K L M N O P Q R S T U V W X Y Z A B C', and the ciphertext 'J D O O L D H V'. A text box explains the encryption process: 'The ciphertext alphabet, being the prerequisite for the encryption, is conducted in the following manner: For each letter, put the third letter character next to the letter in the plaintext alphabet into the ciphertext alphabet. The encryption is made by replacing every letter of the plaintext message by the letter of the ciphertext alphabet.' At the bottom, there are 'Animation controls' (next, forward, pause, etc.) and a 'Step' counter set to 40. A 'Direct selection of an animation step' slider is also present, ranging from 0 to 100.

Animation speed

Scaling of visualisation

Animation controls
(next, forward, pause,
etc.)

Direct selection of
an animation step

Examples (XIV)

Visualisation of symmetric encryption methods using ANIMAL (2)

■ Visualization of DES encryption

Animal Animation: DES Data Encryption Standard (ECB Modus)

Speed 100%

0 200 400 600 800 1000

Input Block X (64-bit)

Key K (64-bit)

Permuted Input

1	0	1	0	0	1	0	0
1	0	1	1	1	0	1	1
0	0	0	1	1	0	1	1
1	0	1	0	0	0	1	0
0	1	0	1	1	0	1	0
1	0	0	1	0	0	1	0
1	1	0	0	0	0	1	0
1	1	1	0	0	0	0	1

K'

0	1	1	1	1	1	1	1
0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	1
1	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	1
0	0	1	1	0	0	0	1

PC2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

K[1]

0	1	1	1	0	1
1	1	1	1	1	1
1					

Step: 165

0 20 40 60 80 100

Animal Animation: DES Data Encryption Standard (ECB Modus)

Speed 100%

0 200 400 600 800 1000

Function f :

110110 001010 110110 010100 000101 100110 101001 010011

B[1] B[2] B[3] B[4] B[5] B[6] B[7] B[8]

$10 = 1 \times 2^1 + 0 \times 2^0 = 2$ $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

S-Box 1:

column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	3

S-Box 8:

column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Step: 294

0 20 40 60 80 100

Examples (XV)

Generation of a message authentication code

Message Authentication Code (MAC)

- Ensures integrity of a message
- Authentication of the message
- Basis: a common key

CrypTool

- Menu „Indiv. Procedures“ / „Hash“ / „Generation of MACs...“

Generation of a MAC in CrypTool

1. Choose a hash function
2. Select MAC variant
3. Enter a key (depending on MAC variant also two keys)
4. Generation of the MAC

Message Authentication Code

By means of a MAC the recipient of a message is able to verify its integrity and the authenticity of its origin (sender). Therefore both parties use a shared secret (symmetric key). To create a MAC, a cryptographic hash function is applied to a combination of the message m and the secret key k . According to the variation chosen below, two different keys k and k' can be used.

1. Choose hash function

- ☒ MD2
- ☐ MD4
- ☐ MD5
- ☐ SHA
- ☐ SHA-1
- ☐ RIPEMD-160

2. Variation (Position of key)

- ☒ $H(k, m)$: in front of message
- ☐ $H(m, k)$: at the back of message
- ☐ $H(k, m, k)$: in front and at the back
- ☐ $H(k, H(k, m))$: double hashing
- ☐ $H(k, m, k')$: different keys

Enter your key (k):
Chiffre

Enter second key (k'):

MAC generated from message and key:
3B D3 95 57 F8 4F C9 C4 1E 0F 6F C2 B3 BC 89 E4

Input to the (outer) hash function (the input is created according to the variation you have chosen above):
ChiffreCrypTool

This is a text file, shown in order to help you to make your first steps with CrypTool.

1) One thing you can do e.g. is to encrypt this file with the Caesar algorithm (via the menu "Crypt \ Classical").

2) The best overview about all features of CrypTool is offered by the starting page of the Windows online help which contains links to all relevant functions.

4. Close Generate MAC

Examples (XVI)

Hash Demo

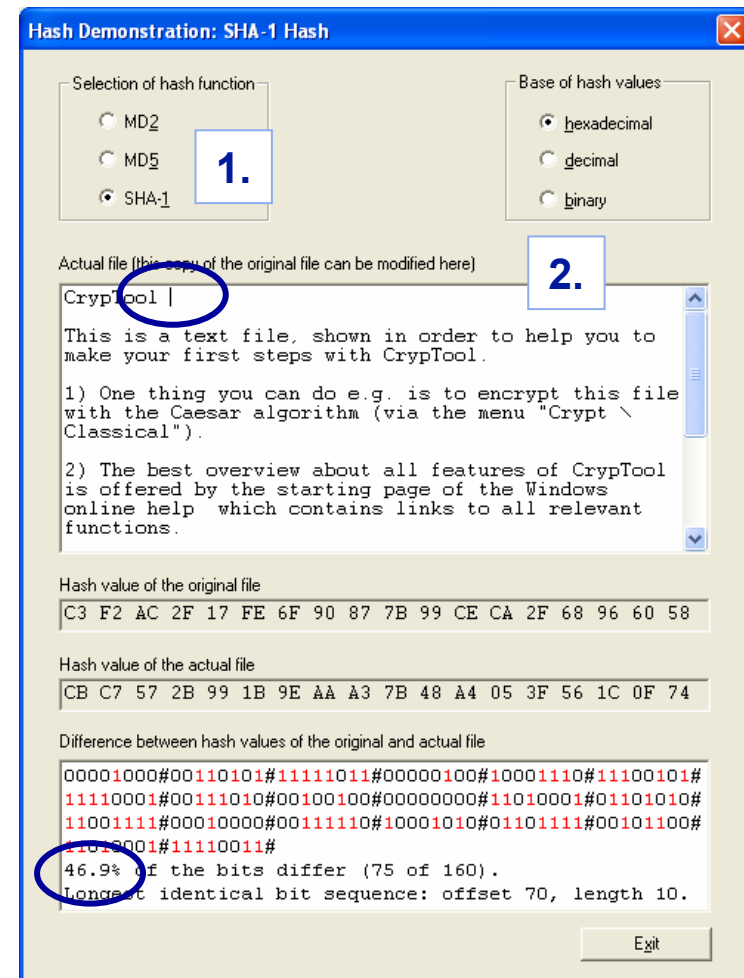
Sensitivity of hash functions to plaintext modifications

1. Select a hash function
2. Modification of characters in plaintext

Example:

Entering a blank after „CrypTool“ in the example text results in a 46,9% change of the bits of the generated hash value.

A good hash function should react sensitive to even the smallest change within the plaintext – „*Avalanche effect*“ (small change, big impact).





Content

[CrypTool and Cryptography – Overview](#)

[CrypTool features](#)

[Examples](#)

Project / Outlook / Contact

Future CrypTool development

Planned after release 1.4.00 (see readme file)

- Mass pattern search
- ECC-Demo, ECC-AES hybrid encryption, ...
- Visualisation of S/MIME and OpenPGP
- Re-design of CrypTool in Java
- Port of C++ version to WPF / Vista
- Integration of crypto library crypto++ from Wei Dai

Future plans (see readme file)

- Visualisation of protocols (e.g. Kerberos)
- Visualisation of attacks on these protocols
- Visualisation of the SSL protocol
- Demonstration of visual cryptography
- Command line interface for batch processing
- Additional parameters for existing methods / algorithms
- Port of C++ version to Linux

CrypTool as a framework for own developments

Proposal

- Re-use the comprehensive set of algorithms, included libraries and interface elements as foundation
- Free of charge training in Frankfurt, how to start with CrypTool development
- Advantage: Own code does not „disappear“, but will be maintained

Current development environment: **Microsoft VC++ , Perl, Subversion source code management**

- Until CrypTool 1.3.05: Visual C++ 6.0 only (was available within books for free)
- CrypTool 1.4.00: Visual C++ .net (= VC++ 7.1)(= Visual Studio 2003)
- Description for developers: see readme-source.txt
- Download: Sources and binaries of releases.
To get sources of current betas, please send an eMail.

Future development environment

- For versions after 1.4.00:
 - C++ version: .NET (without MFC) with Visual Studio 2005 Express Edition (free), WPF and Perl
 - Java version: with Eclipse 3.1, SWT (free)
- Considered:
 - C++ version for Linux with Qt 4.x, GCC 4.0 and Perl

CrypTool – request for contribution

- **Every contribution to the project is highly appreciated**
 - Feedback, criticism, suggestions and ideas
 - Integration of additional algorithms, protocols, analysis (consistency and completeness)
 - Development assistance (programming, layout, translation, test)
 - For the C/C++ project as well as for the new Java project !
 - Especially University faculties using CrypTool for educational purposes are invited to contribute to the further development of CrypTool.
 - Significant contributions can be referenced by name (in help, readme, about dialog and on the CrypTool web site).
 - Currently CrypTool is being downloaded more than 2000 times a month (with 1/3 for the English version).

Contact

Bernhard Esslinger

**University of Siegen
University lecturer, Faculty 5, Economics and Business Computing**

**Deutsche Bank AG
Director, IT Security Manager**

esslinger@fb5.uni-siegen.de

**www.cryptool.de
www.cryptool.org
www.cryptool.com**

**additional contacts: See readme within the CrypTool folder
mailing list: cryptool-list@sec.informatik.tu-darmstadt.de**

Additional Literature

- Simon Singh, *“The Codebook”*, 1999, Doubleday [English]
- Simon Singh, *“Geheime Botschaften”*, 2000, Hanser [German]
- U. Ulfkotte, *“Wirtschaftsspionage”*, 2001, Goldmann [German]
- Claudia Eckert, *“IT-Sicherheit”*, 3rd edition, 2004, Oldenbourg [German]
- A. Beutelspacher / J. Schwenk / K.-D. Wolfenstetter, *“Moderne Verfahren der Kryptographie”*, 5th edition, 2004, Vieweg [German]
- [HAC] Menezes, van Oorschot, Vanstone, *“Handbook of Applied Cryptography”*, 1996, CRC Press
- Van Oorschot, Wiener, *“Parallel Collision Search with Application to Hash Functions and Discrete Logarithms”*, 1994
- Additional cryptography literature
(e.g. by Wätjen, Buchmann, Salomaa, Brands, Schneier, Shoup, ...)
- Importance of cryptography in the broader context of IT security and risk management
 - See e.g. Kenneth C. Laudon / Jane P. Laudon / Detlef Schoder, *“Wirtschaftsinformatik”*, 2005, Pearson, chapter 14 [German]
 - See Wikipedia (http://en.wikipedia.org/wiki/Risk_management)